Timed pushdown automata and branching vector addition systems

Sławomir Lasota
University of Warsaw

joint work with Lorenzo Clemente, Filip Mazowiecki and Ranko Lazic

AVERTS 2016, Chennai

Definable sets

offer a right setting for timed models of computation, like timed automata, or timed pushdown automata.

Definable sets

offer a right setting for timed models of computation, like timed automata, or timed pushdown automata.

Definable PDA

have decidable non-emptiness problem, by reduction to an extension of BVASS in dimension 1.

Plan

- **Motivation**
- Definable NFA
- Definable PDA
- The core problem: equations over sets of integers
- Branching vector addition systems in dimension 1

Time domain

- reals
- rationals
- integers

$\left.\begin{array}{l}\\ \\ \end{array}\right\}$ dense time

discrete time

any choice of time domain is fine

Time domain

- reals
- rationals
- integers

dense time

discrete time

any choice of time domain is fine

Time domain

- reals
- rationals
- integers

dense time

discrete time

any choice of time domain is fine

No restriction to non-negative!

Time domain

- reals
- rationals
- integers

dense time

discrete time

any choice of time domain is fine

No restriction to non-negative!

Let input alphabet be reals

Time domain

- reals
- rationals
- integers

dense time

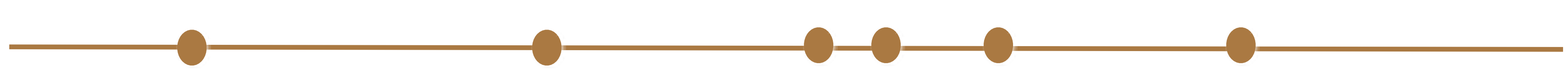
discrete time

any choice of time domain is fine

No restriction to non-negative!

Let input alphabet be reals

Timed automata assume monotonic input words :



Timed automata [Alur, Dill 1990]

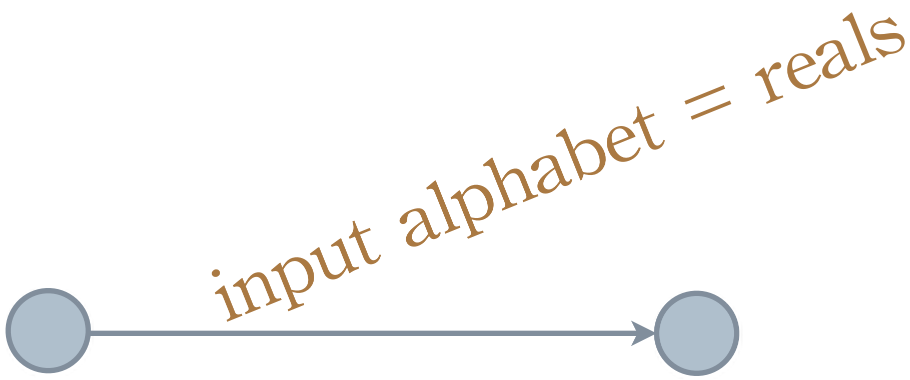
with uninitialized clocks

Timed automata [Alur, Dill 1990]
with uninitialized clocks

?

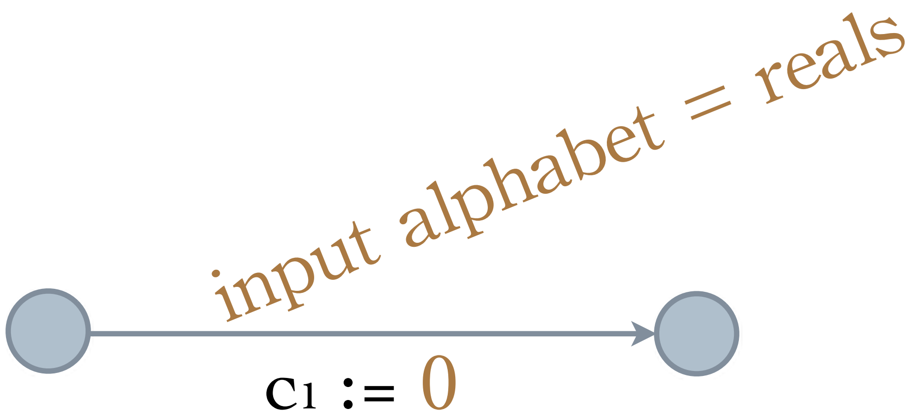
Timed automata [Alur, Dill 1990]

with uninitialized clocks



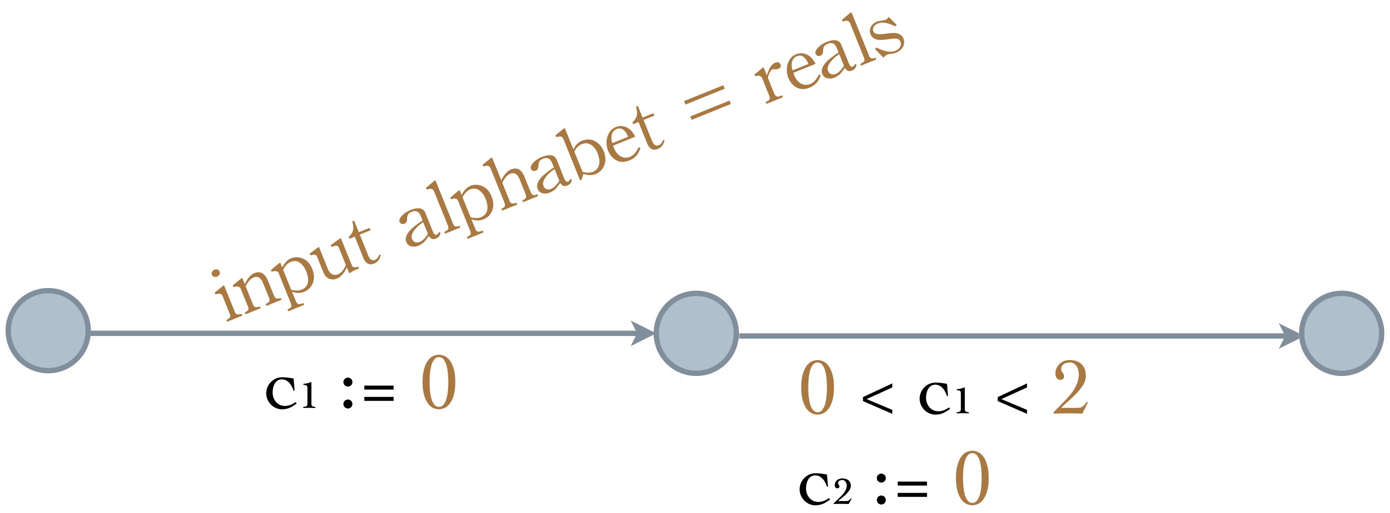
Timed automata [Alur, Dill 1990]

with uninitialized clocks



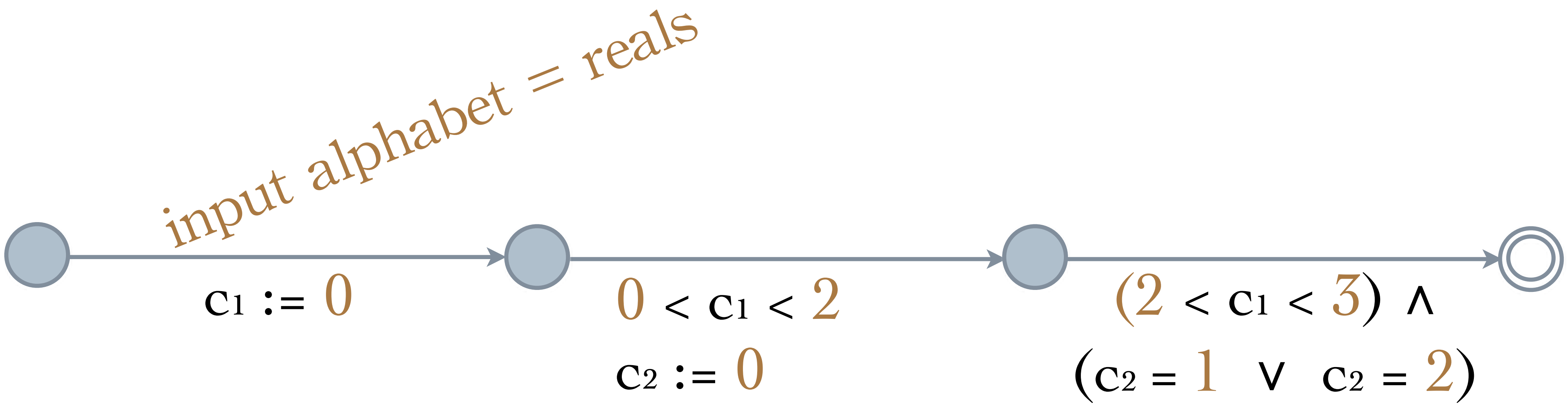
Timed automata [Alur, Dill 1990]

with uninitialized clocks



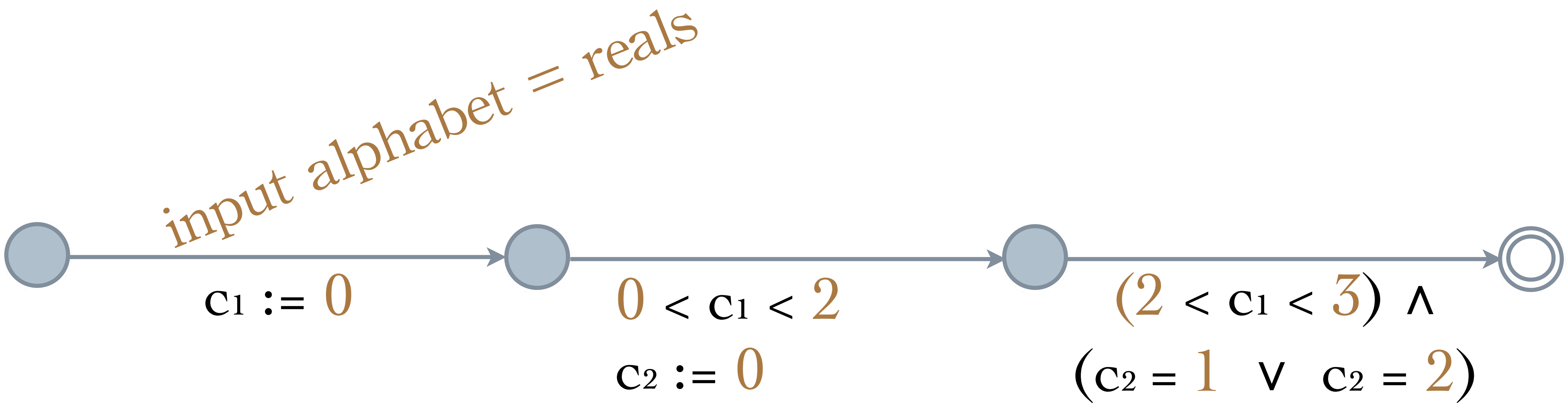
Timed automata [Alur, Dill 1990]

with uninitialized clocks



Timed automata [Alur, Dill 1990]

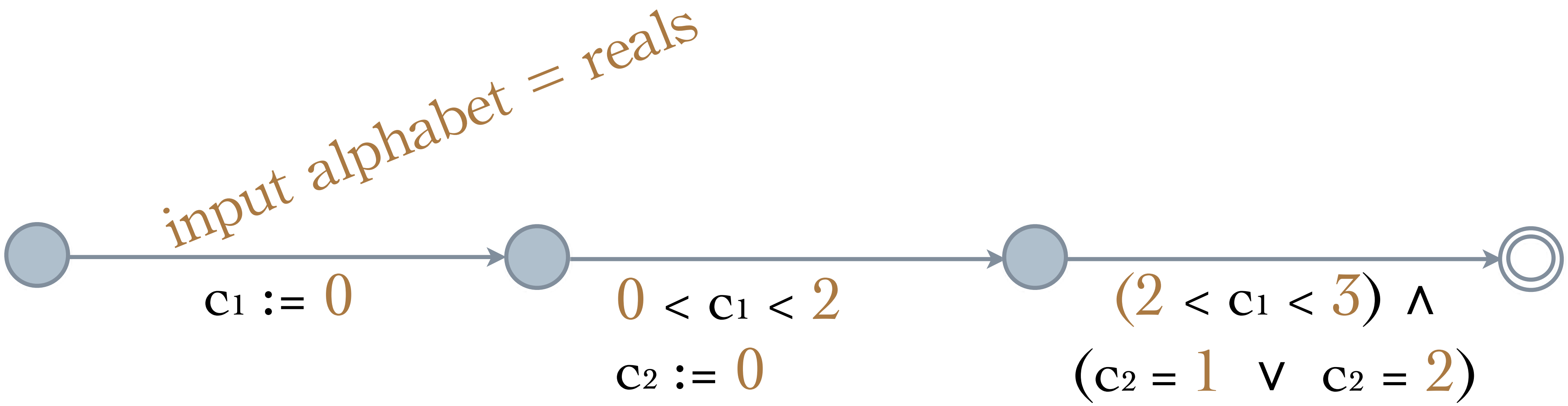
with uninitialized clocks



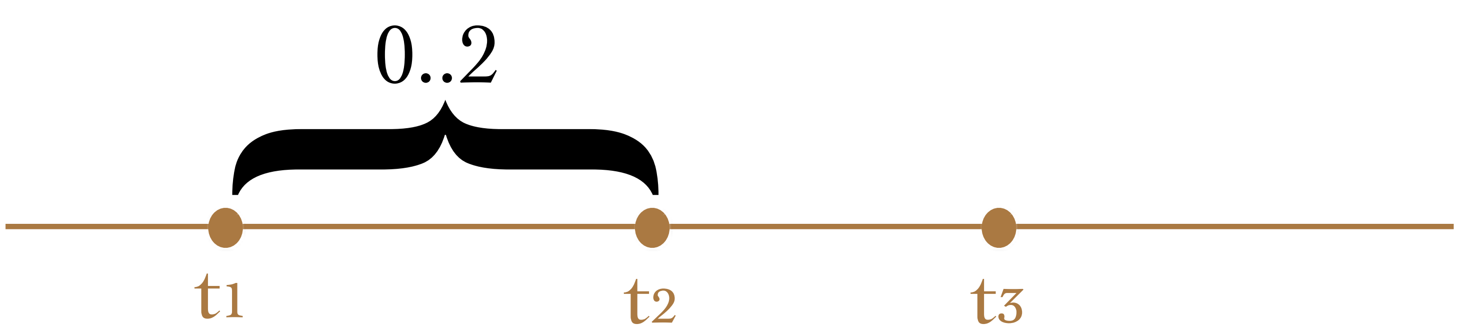
the automaton accepts words $t_1\, t_2\, t_3 \in \mathbb{R}^3$ such that

Timed automata [Alur, Dill 1990]

with uninitialized clocks

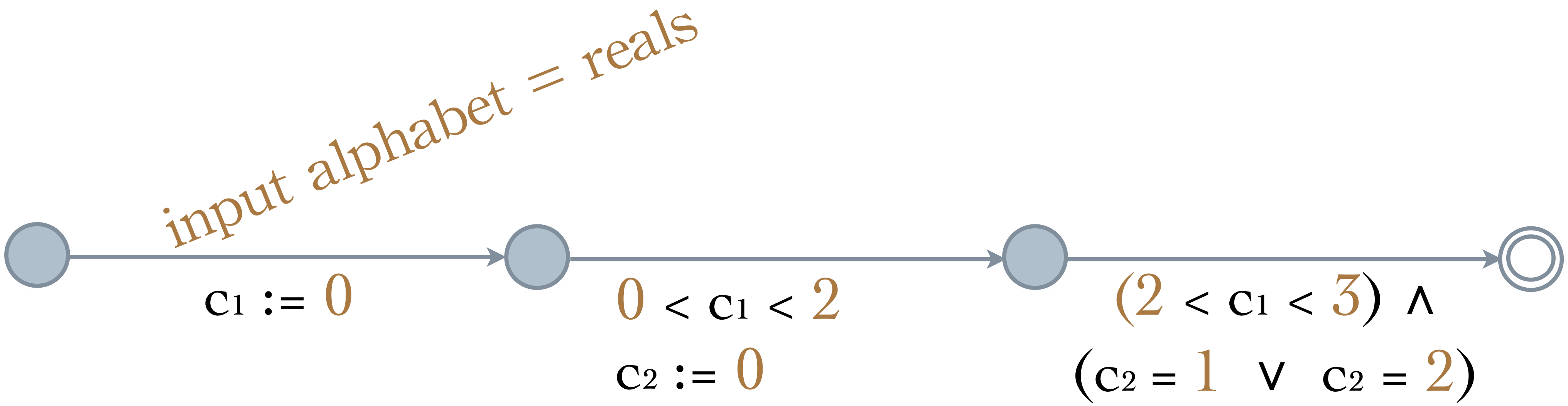


the automaton accepts words $t_1\ t_2\ t_3 \in R^3$ such that

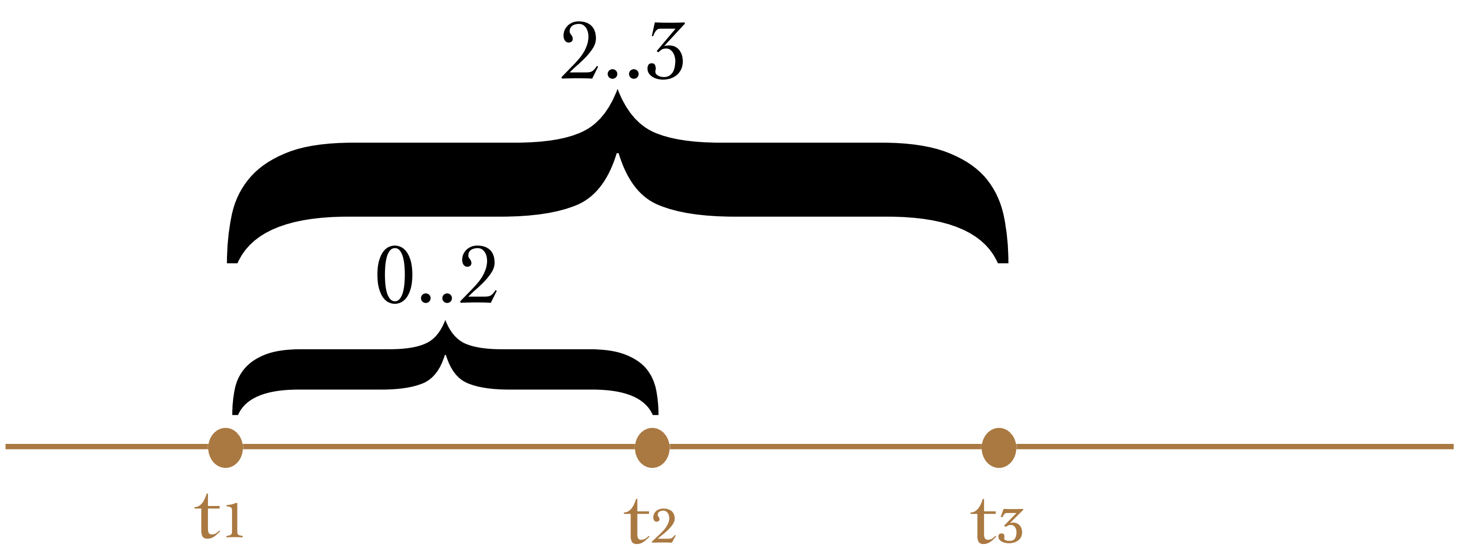


Timed automata [Alur, Dill 1990]

with uninitialized clocks

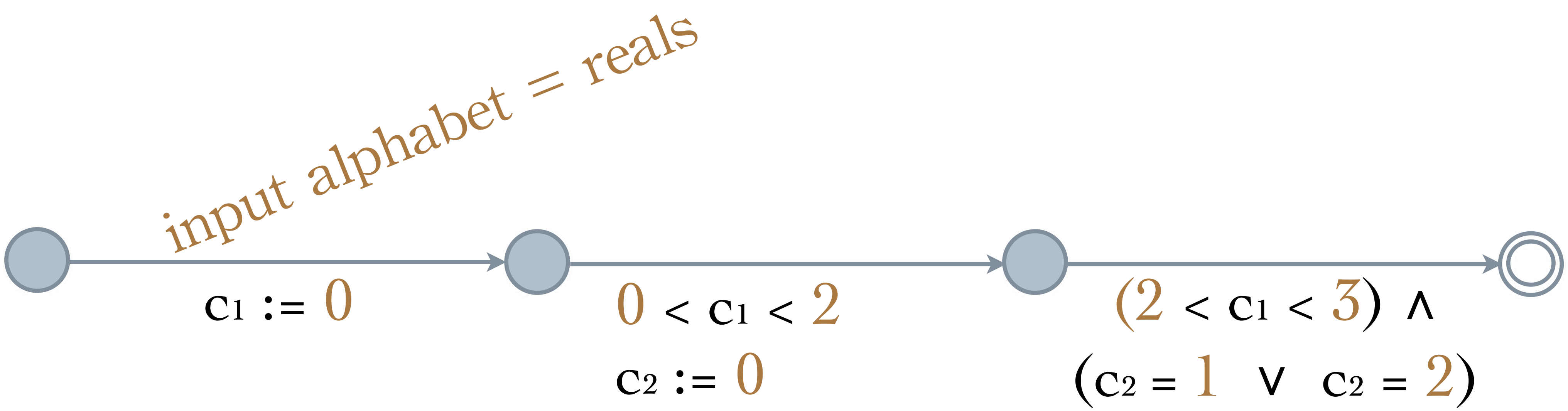


the automaton accepts words $t_1\ t_2\ t_3 \in R^3$ such that

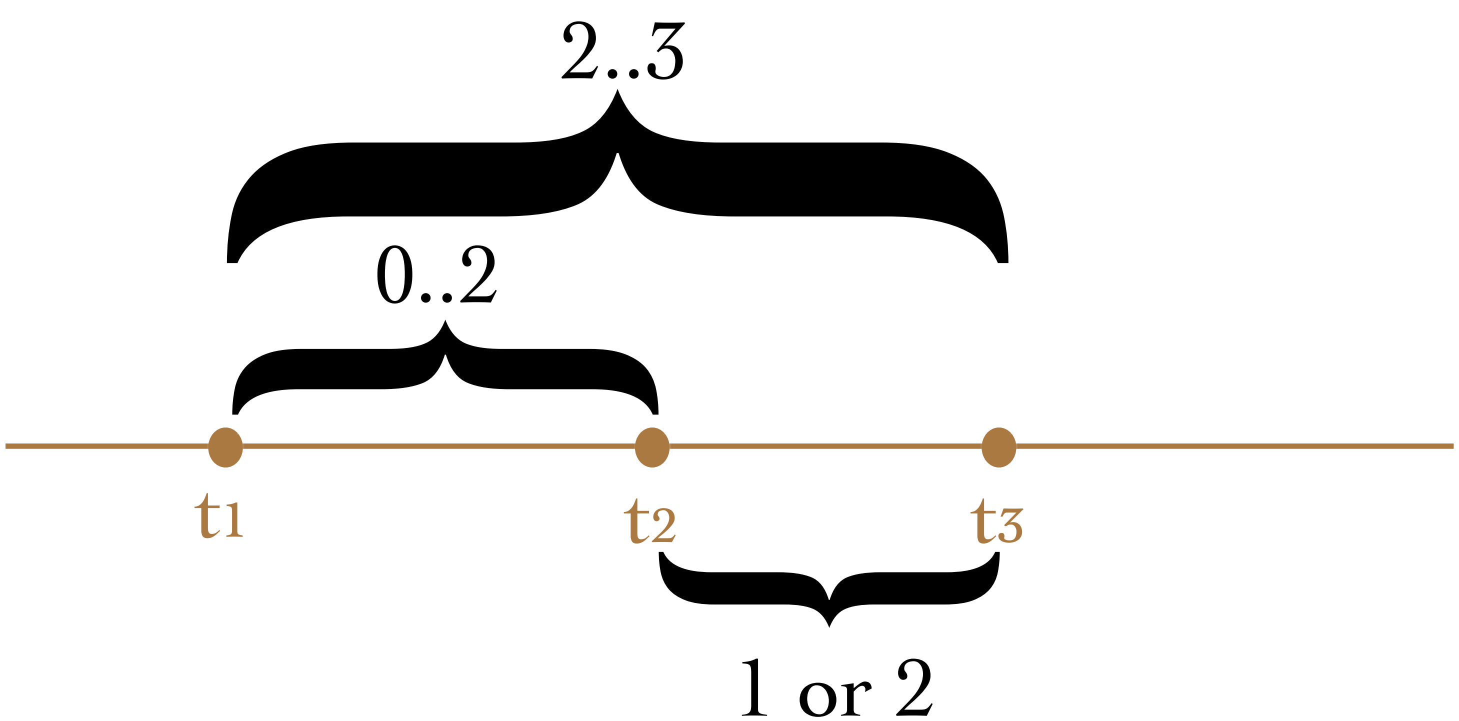


Timed automata [Alur, Dill 1990]

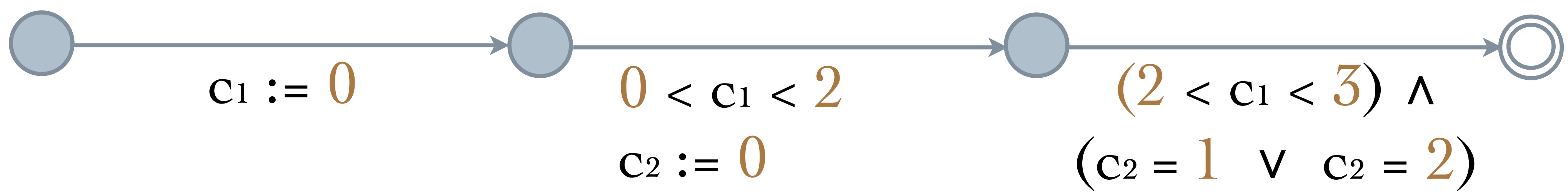
with uninitialized clocks

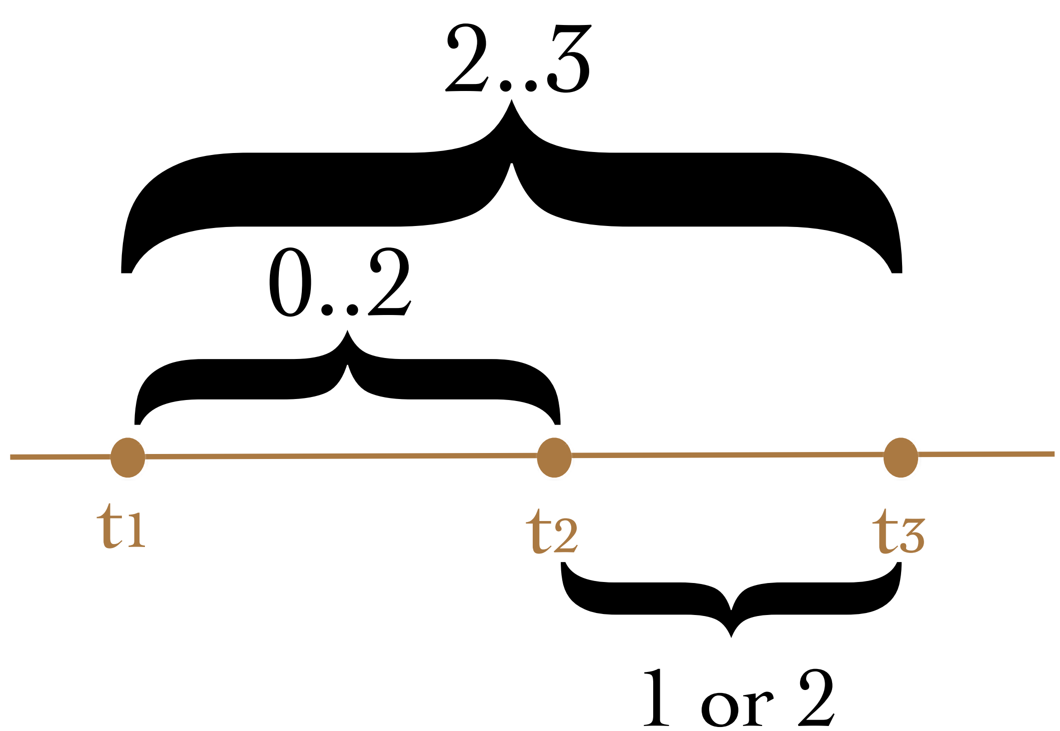


the automaton accepts words $t_1\ t_2\ t_3 \in R^3$ such that

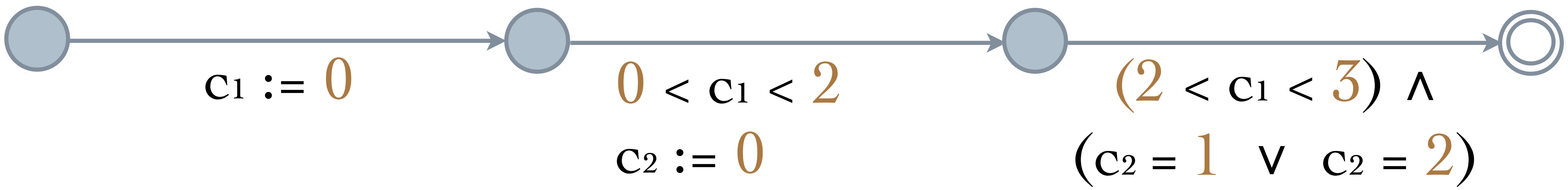


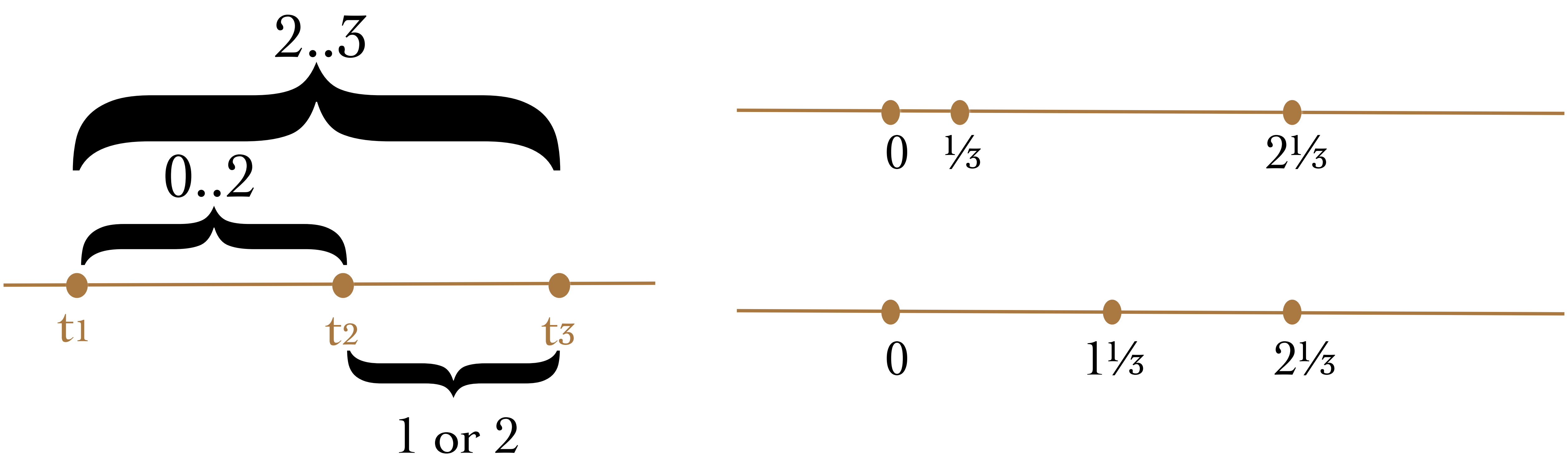
Deterministic timed automata don't minimize



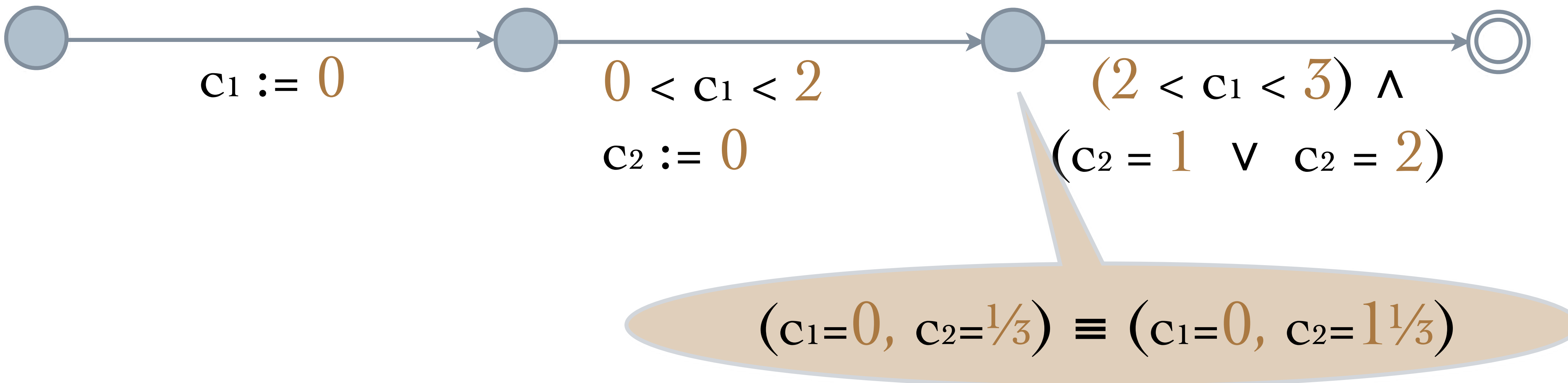


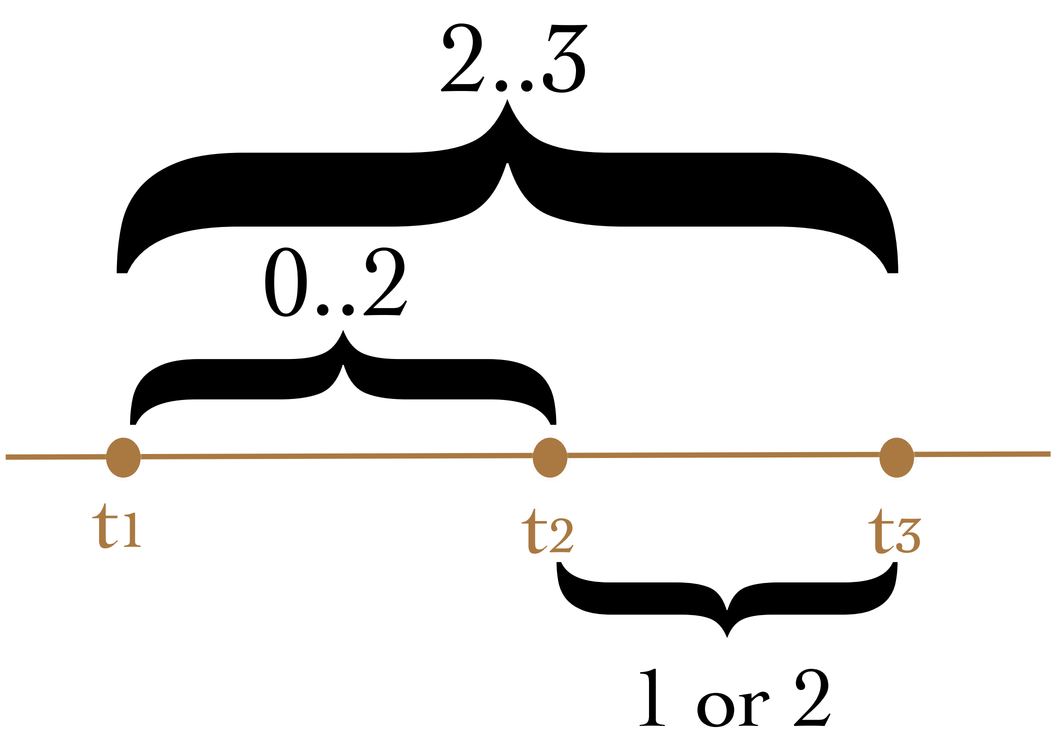
Deterministic timed automata don't minimize

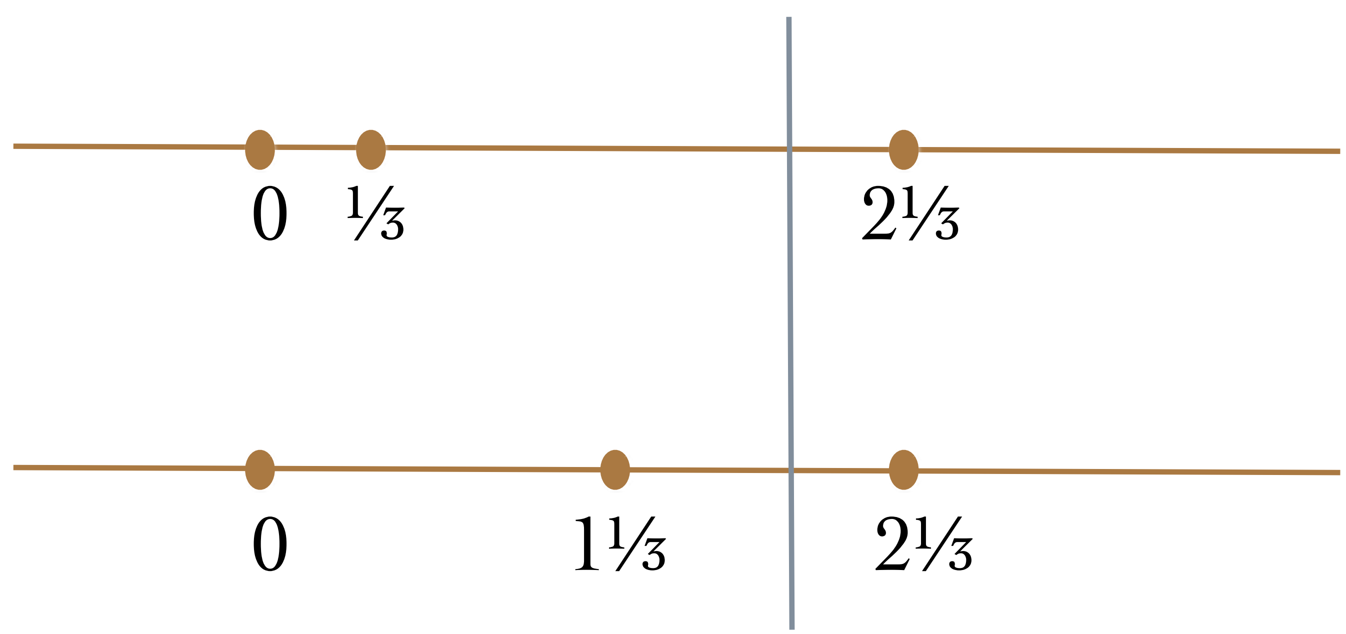




Deterministic timed automata don't minimize







Towards timed pushdown automata

Towards timed pushdown automata

- timed automata [Alur, Dill 1990]

Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]

finite stack alphabet

Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]

  finite stack alphabet

- dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]

  - clocks can be pushed onto stack
  - the emptiness problem EXPTIME-c

Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
- dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]
- recursive timed automata [Trivedi, Wojtczak 2010], [Benerecetti, Minopoli, Peron 2010]

finite stack alphabet

- clocks can be pushed onto stack
- the emptiness problem EXPTIME-c

Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
- dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]

finite stack alphabet

- clocks can be pushed onto stack
- the emptiness problem EXPTIME-c

Theorem 1: [Clemente, L. 2015]
Dense-timed pushdown automata are expressively equivalent to pushdown timed automata.

Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
- dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]

finite stack alphabet

- clocks can be pushed onto stack
- the emptiness problem EXPTIME-c

Theorem 1: [Clemente, L. 2015]
Dense-timed pushdown automata are expressively equivalent to pushdown timed automata.

An accidental combination of

- stack discipline
- monotonicity of time
- syntactic restrictions

- do not invent a new definition

- do not invent a new definition
- re-interpret a classical definition in **definable** sets, with finiteness relaxed to **orbit-finiteness**

- do not invent a new definition
- re-interpret a classical definition in **definable** sets, with finiteness relaxed to **orbit-finiteness**

- alphabet A
- states Q
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

- do not invent a new definition
- re-interpret a classical definition in **definable** sets, with finiteness relaxed to **orbit-finiteness**

- alphabet A
- states Q
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

definable

- do not invent a new definition
- re-interpret a classical definition in **definable** sets, with finiteness relaxed to **orbit-finiteness**

$$\left.\begin{array}{l}\left.\begin{array}{l}\bullet\ \text{alphabet } A\\ \bullet\ \text{states } Q\end{array}\right\}\text{orbit-finite}\\ \bullet\ \text{transitions } \delta \subseteq Q \times A \times Q\\ \bullet\ I, F \subseteq Q\end{array}\right\}\text{definable}$$

In search of lost definition

- Motivation
- **Definable NFA**
- Definable PDA
- The core problem: equations over sets of integers
- Branching vector addition systems in dimension 1

In search of lost definition

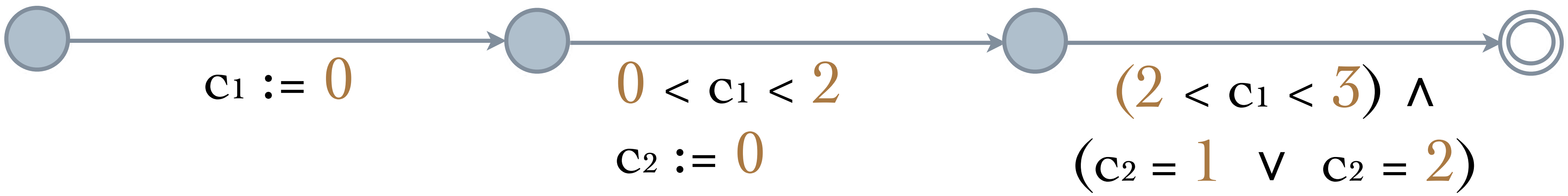
- Motivation
- **Definable NFA**
- Definable PDA
- The core problem: equations over sets of integers
- Branching vector addition systems in dimension 1

NFA re-interpreted in definable sets

Timed automata are register automata

with uninitialized clocks

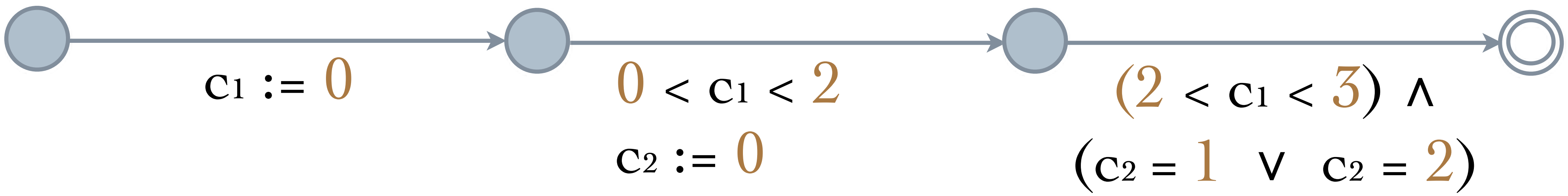
[Bojańczyk, L. 2012]

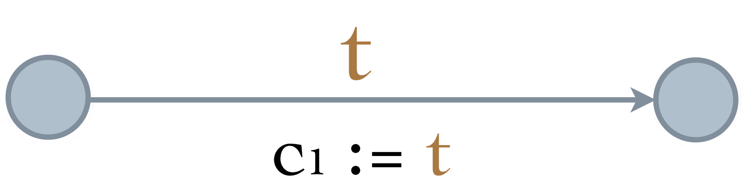


Timed automata are register automata

with uninitialized clocks

[Bojańczyk, L. 2012]

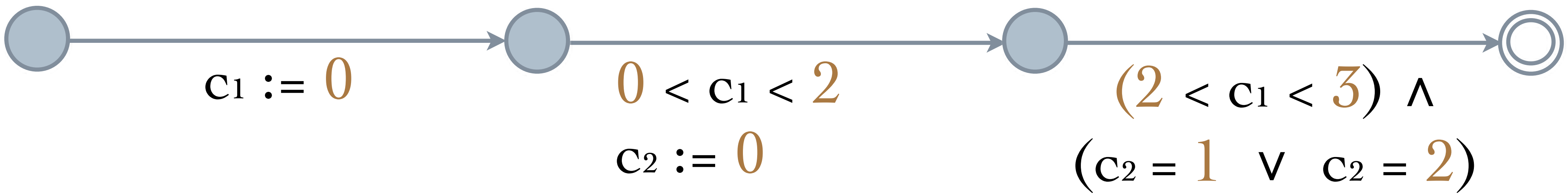


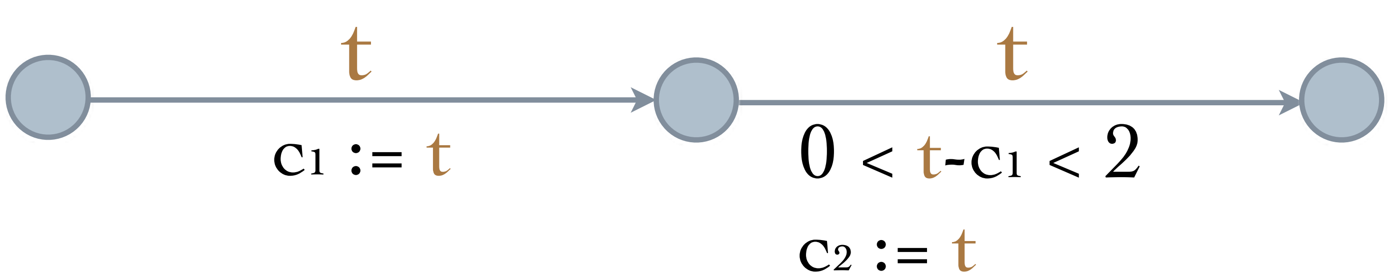


Timed automata are register automata

with uninitialized clocks

[Bojańczyk, L. 2012]

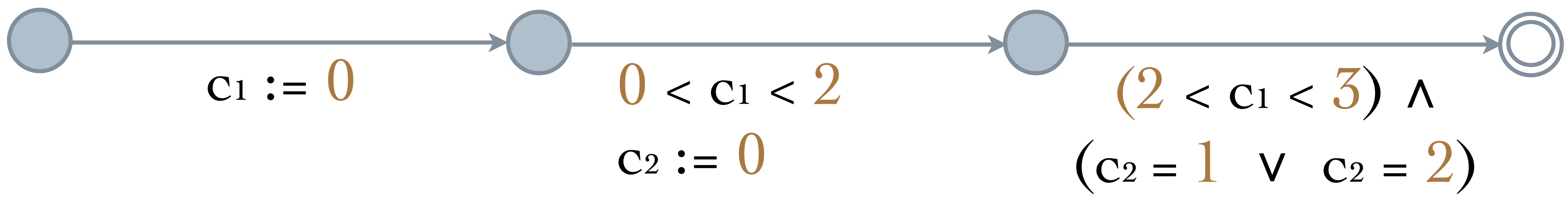


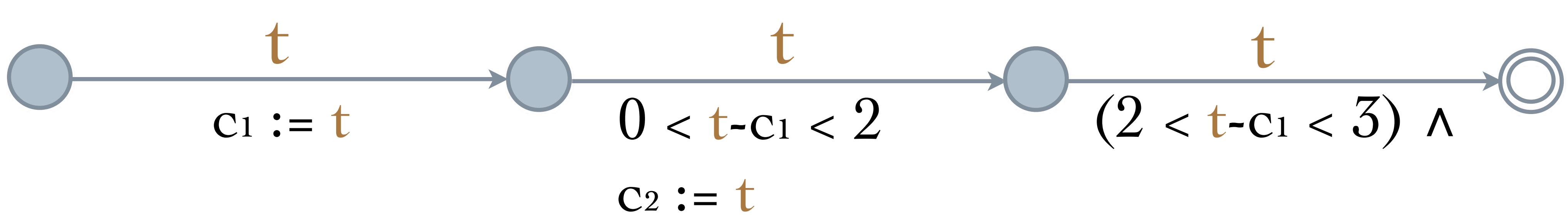


Timed automata are register automata

with uninitialized clocks

[Bojańczyk, L. 2012]

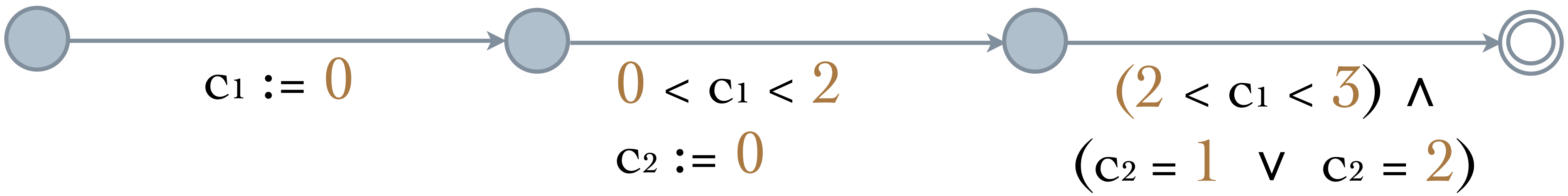


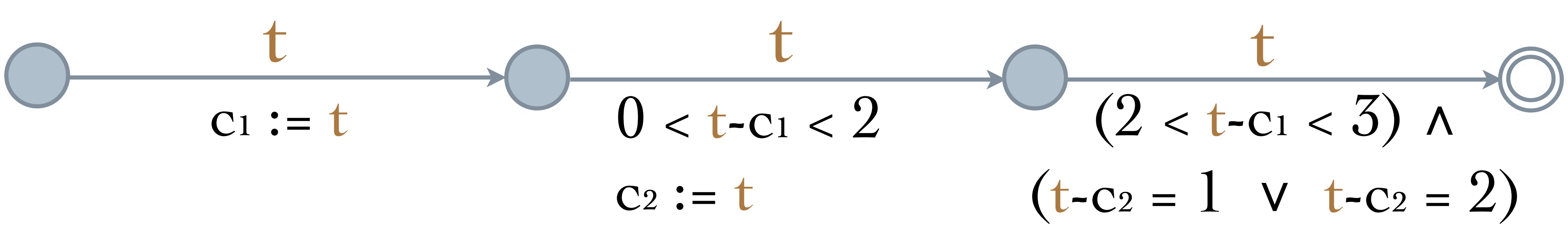


Timed automata are register automata

with uninitialized clocks

[Bojańczyk, L. 2012]

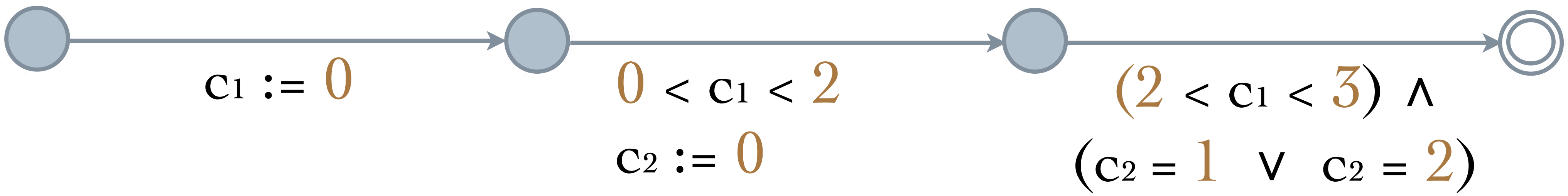


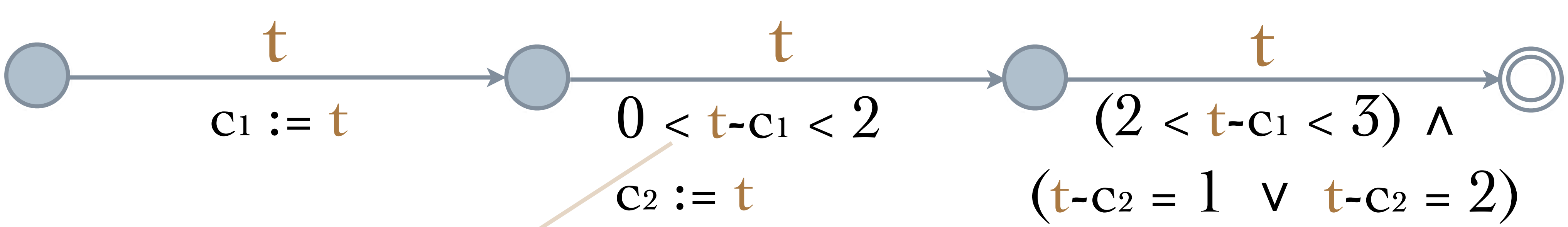


Timed automata are register automata

with uninitialized clocks

[Bojańczyk, L. 2012]





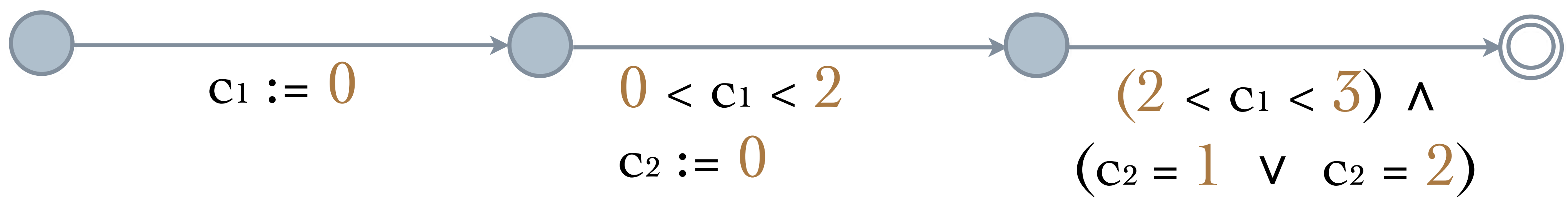
the guards use the structure $(\mathbb{R}, <, +1)$

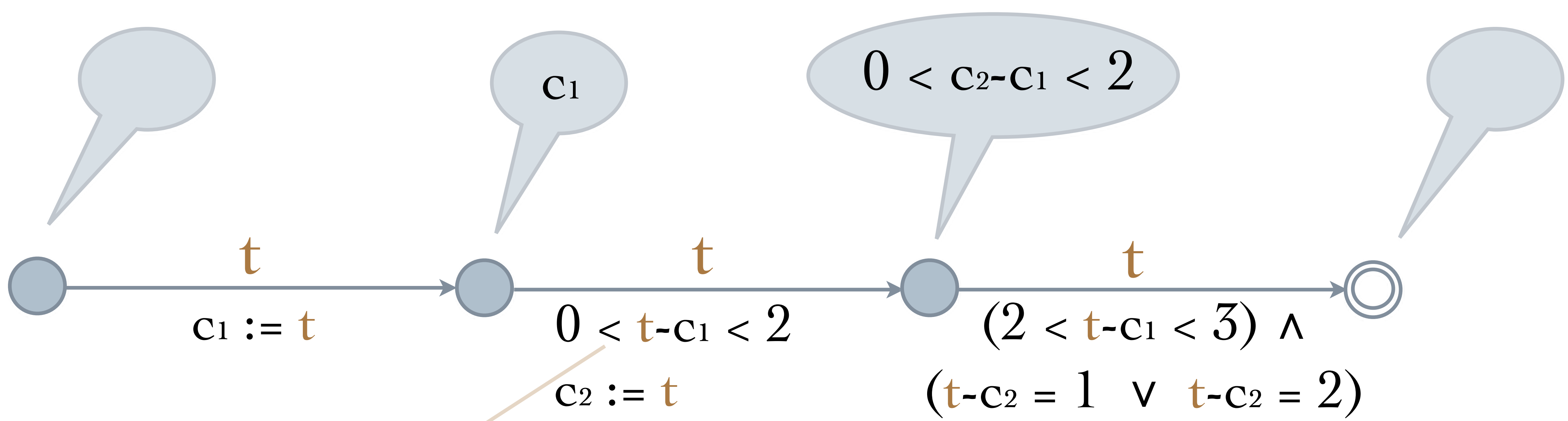
e.g. $0 < t - c_1 < 2$ iff $c_1 < t < c_1 + 2$

Timed automata are register automata

with uninitialized clocks

[Bojańczyk, L. 2012]





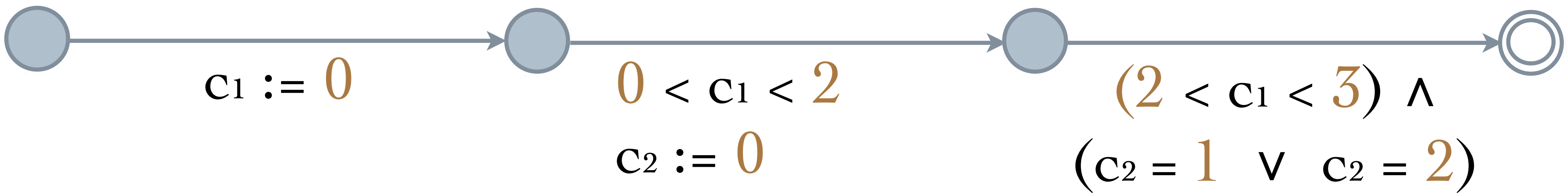
the guards use the structure $(\mathbb{R}, <, +1)$

e.g. $0 < t - c_1 < 2$ iff $c_1 < t < c_1 + 2$

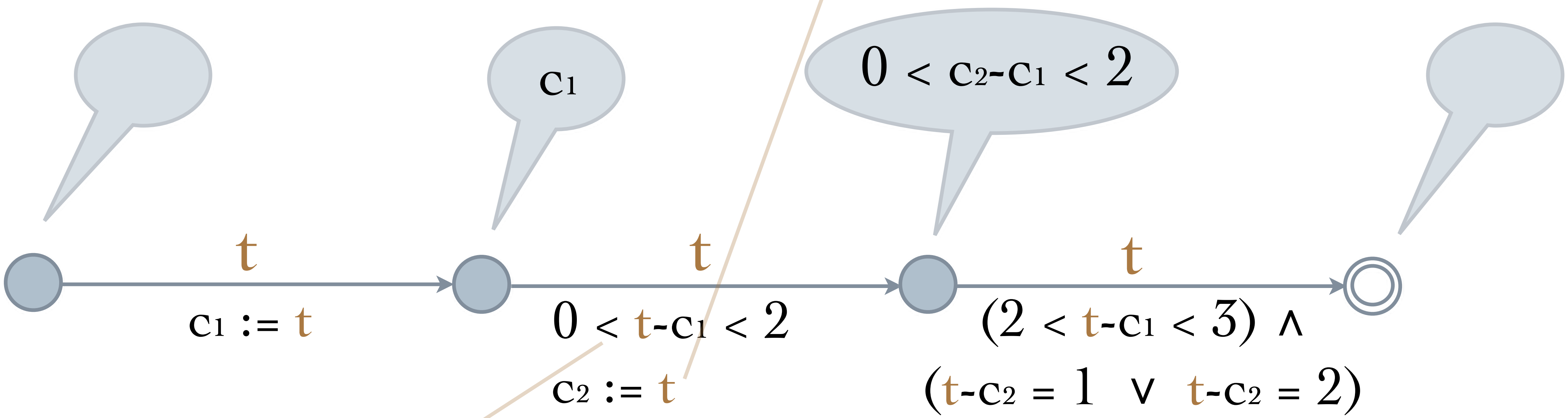
Timed automata are register automata

with uninitialized clocks

[Bojańczyk, L. 2012]



the only modifications of a clock: $c := t$



the guards use the structure $(\mathbb{R}, <, +1)$

e.g. $0 < t - c_1 < 2$ iff $c_1 < t < c_1 + 2$

(<, +1)-definable sets

dimension

FO(<, +1) formula $\phi(x_1, \ldots, x_n)$ defines a subset of n-tuples of reals, for instance

$$\phi(x_1, x_2) \quad \equiv \quad \exists x_3 \ (x_1 < x_3 \ \wedge \ x_2 = x_3 + 3)$$

definable sets

dimension

FO(<, +1) formula $\phi(x_1, \ldots, x_n)$ defines a subset of n-tuples of reals, for instance

$$\phi(x_1, x_2) \quad \equiv \quad \exists x_3 \ (x_1 < x_3 \ \wedge \ x_2 = x_3 + 3)$$

definable sets

dimension

FO(<, +1) formula $\phi(x_1, \ldots, x_n)$ defines a subset of n-tuples of reals, for instance

$$\phi(x_1, x_2) \quad \equiv \quad \exists x_3 \; (x_1 < x_3 \;\wedge\; x_2 = x_3 + 3)$$

FO(<, +1) = QF(<, +1) =

definable sets

dimension

FO(<, +1) formula $\phi(x_1, \ldots, x_n)$ defines a subset of n-tuples of reals, for instance

$$\phi(x_1, x_2) \quad \equiv \quad \exists x_3 \; (x_1 < x_3 \; \wedge \; x_2 = x_3 + 3)$$

FO(<, +1) = QF(<, +1) = $\displaystyle\bigvee_{\text{finite}} \underbrace{\bigwedge_{\text{finite}} x_i - x_j \in I_{ij}}_{\text{zone}}$

definable sets

dimension

FO(<, +1) formula $\phi(x_1, \ldots, x_n)$ defines a subset of n-tuples of reals, for instance

$$\phi(x_1, x_2) \quad \equiv \quad \exists x_3 \ (x_1 < x_3 \ \wedge \ x_2 = x_3 + 3)$$

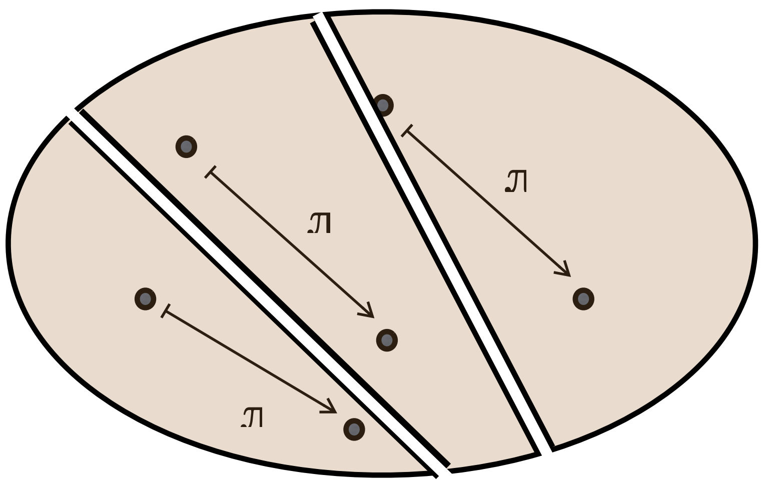
$$\text{FO}(<, +1) = \text{QF}(<, +1) = \bigvee_{\text{finite}} \underbrace{\bigwedge_{\text{finite}} x_i - x_j \in I_{ij}}_{\text{zone}}$$

for instance:

$$\phi(x_1, x_2) \quad \equiv \quad x_1 + 3 < x_2 \quad \equiv \quad x_2 - x_1 \in (3, \infty)$$

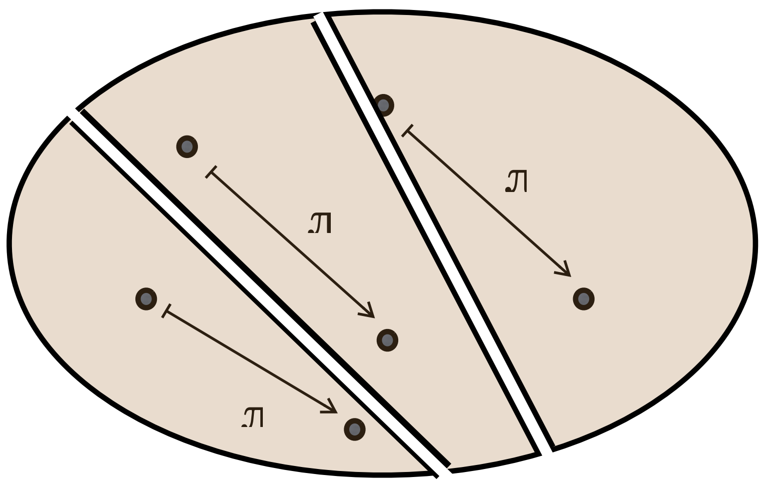
Orbit-finiteness

Automorphisms π of (R, <, +1) act on a definable set thus splitting it into orbits.



Orbit-finiteness

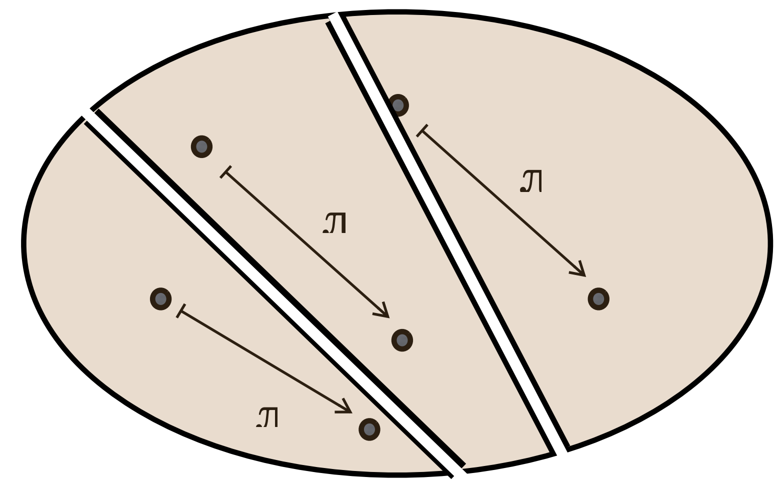
Automorphisms π of (R, <, +1) act on a definable set thus splitting it into orbits.



For instance, (-1, ⅓) and (3, 4⅓) and (1⅓, 3) are in the same orbit.

Orbit-finiteness

Automorphisms π of (R, <, +1) act on a definable set thus splitting it into orbits.



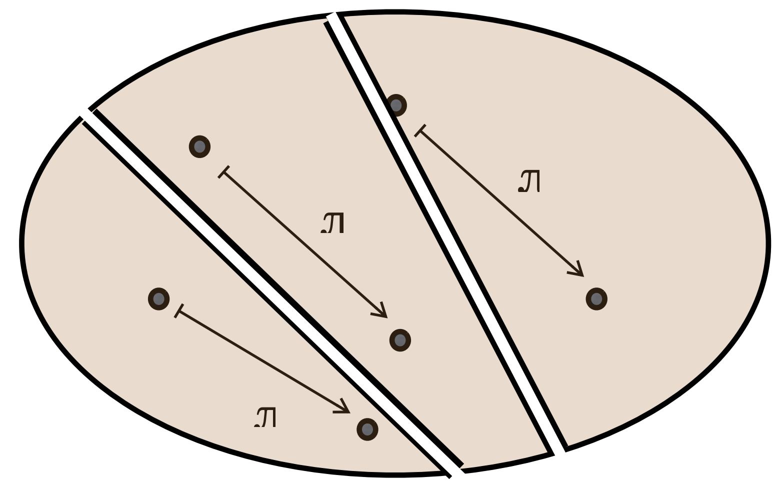
For instance, (-1, ⅓) and (3, 4⅓) and (1⅓, 3) are in the same orbit.

Example:

$x_1 + 3 < x_2 \quad \equiv \quad x_2 - x_1 \in (3, \infty)$ orbit-infinite

Orbit-finiteness

Automorphisms π of (R, <, +1) act on a definable set thus splitting it into orbits.



For instance, (-1, ⅓) and (3, 4⅓) and (1⅓, 3) are in the same orbit.

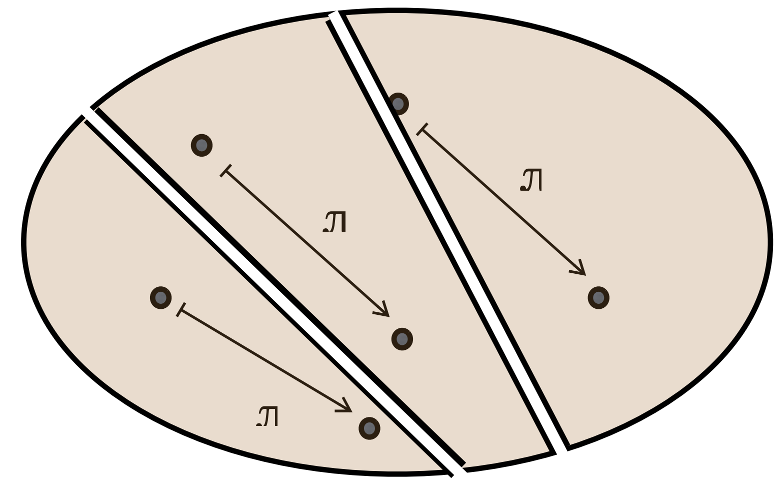
Example:

$$x_1 + 3 < x_2 \quad \equiv \quad x_2 - x_1 \in (3, \infty)$$ orbit-infinite

$$x_1 + 3 < x_2 \leq x_1 + 7 \quad \equiv \quad x_2 - x_1 \in (3, 7]$$ orbit-finite

Orbit-finiteness

Automorphisms π of (R, <, +1) act on a definable set thus splitting it into orbits.



For instance, (-1, ⅓) and (3, 4⅓) and (1⅓, 3) are in the same orbit.

Example:

$x_1 + 3 < x_2 \quad \equiv \quad x_2 - x_1 \in (3, \infty)$ orbit-infinite

$x_1 + 3 < x_2 \leq x_1 + 7 \quad \equiv \quad x_2 - x_1 \in (3, 7]$ orbit-finite

A definable set is orbit-finite
iff
it is defined using bounded intervals only

Definable NFA

- alphabet A
- states Q
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

Definable NFA

- alphabet A
- states Q
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

(<, +1)-definable

Definable NFA

- alphabet A $\phi_A(x_1, \ldots, x_n)$
- states Q $\phi_Q(x_1, \ldots, x_m)$
- transitions δ ⊆ Q × A × Q $\phi_\delta(x_1, \ldots, x_{m+n+m})$
- I, F ⊆ Q $\phi_I(x_1, \ldots, x_m),\ \phi_F(x_1, \ldots, x_m)$

Definable NFA

- alphabet A
- states Q
- transitions $\delta \subseteq Q \times A \times Q$
- I, $F \subseteq Q$

$\}$ (<, +1)-definable

$$\phi_A(x_1, \ldots, x_n)$$
$$\phi_Q(x_1, \ldots, x_m)$$
$$\phi_\delta(x_1, \ldots, x_{m+n+m})$$
$$\phi_I(x_1, \ldots, x_m),\ \phi_F(x_1, \ldots, x_m)$$

Definable NFA

- alphabet A
- states Q
- transitions δ ⊆ Q × A × Q
- I, F ⊆ Q

(<, +1)-definable

$$\phi_A(x_1, \ldots, x_n)$$
$$\phi_Q(x_1, \ldots, x_m)$$
$$\phi_\delta(x_1, \ldots, x_{m+n+m})$$
$$\phi_I(x_1, \ldots, x_m),\ \phi_F(x_1, \ldots, x_m)$$

Runs, acceptance, language recognized, etc. are defined exactly as for classical NFA!

Definable NFA

- alphabet A
- states Q
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

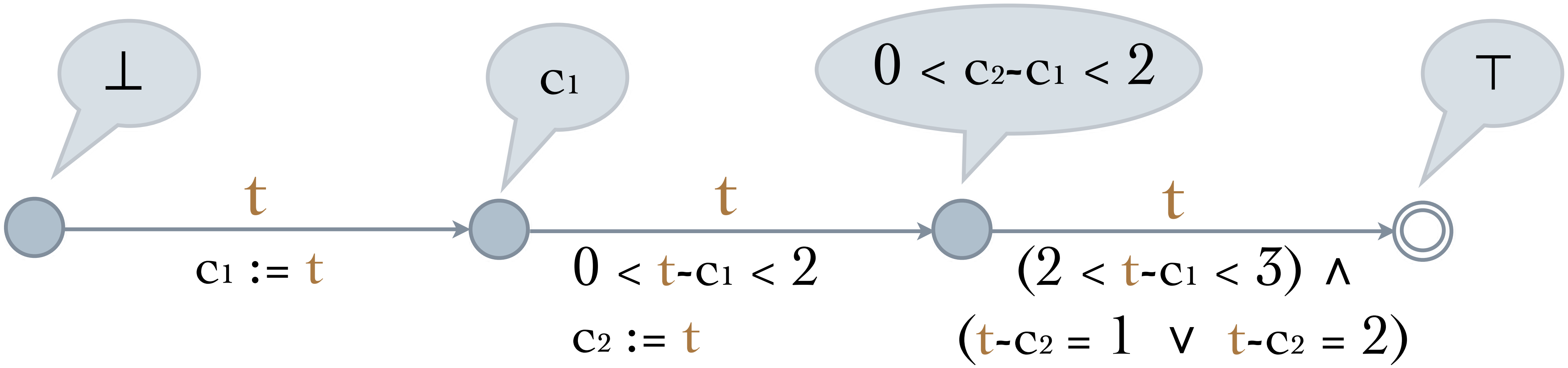
orbit-finite

(<, +1)-definable

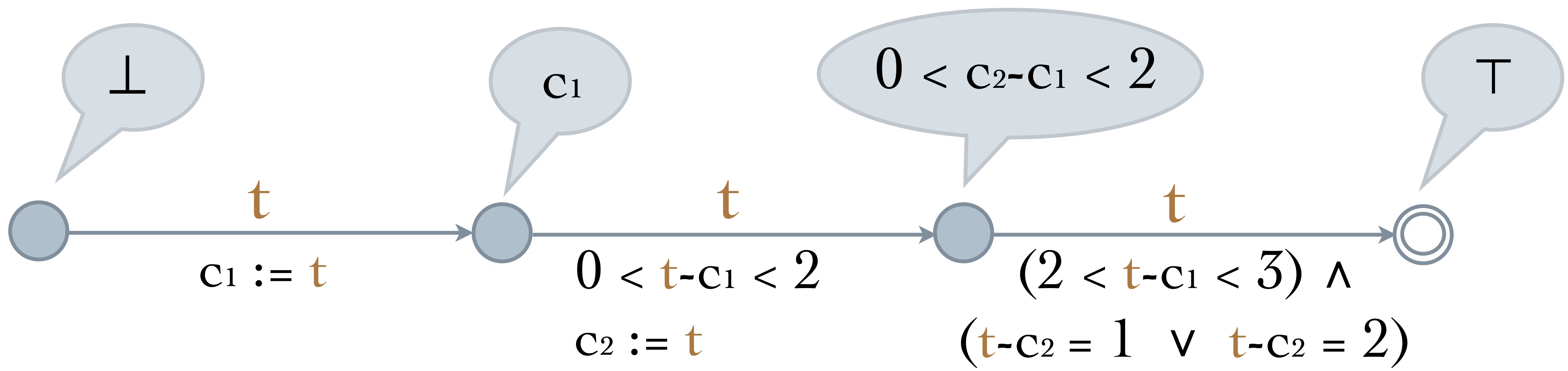
$$\phi_A(x_1, \ldots, x_n)$$
$$\phi_Q(x_1, \ldots, x_m)$$
$$\phi_\delta(x_1, \ldots, x_{m+n+m})$$
$$\phi_I(x_1, \ldots, x_m),\ \phi_F(x_1, \ldots, x_m)$$

Runs, acceptance, language recognized, etc. are defined exactly as for classical NFA!

Register automata = definable NFA

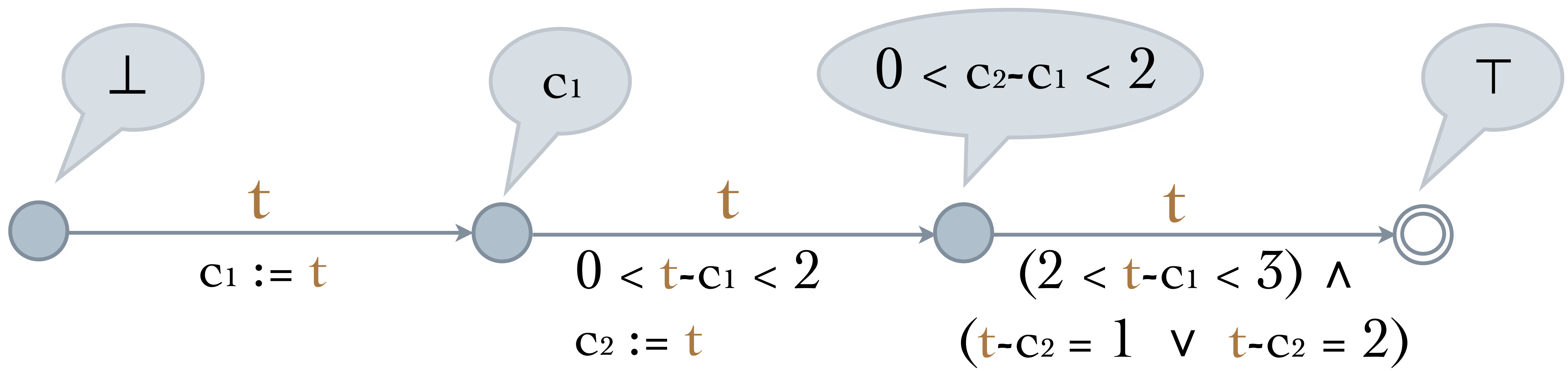


Register automata = definable NFA



states: $Q = \{\bot\} \cup R \cup \{ (c_1, c_2) \in R \times R : 0 < c_2 - c_1 < 2 \} \cup \{\top\}$

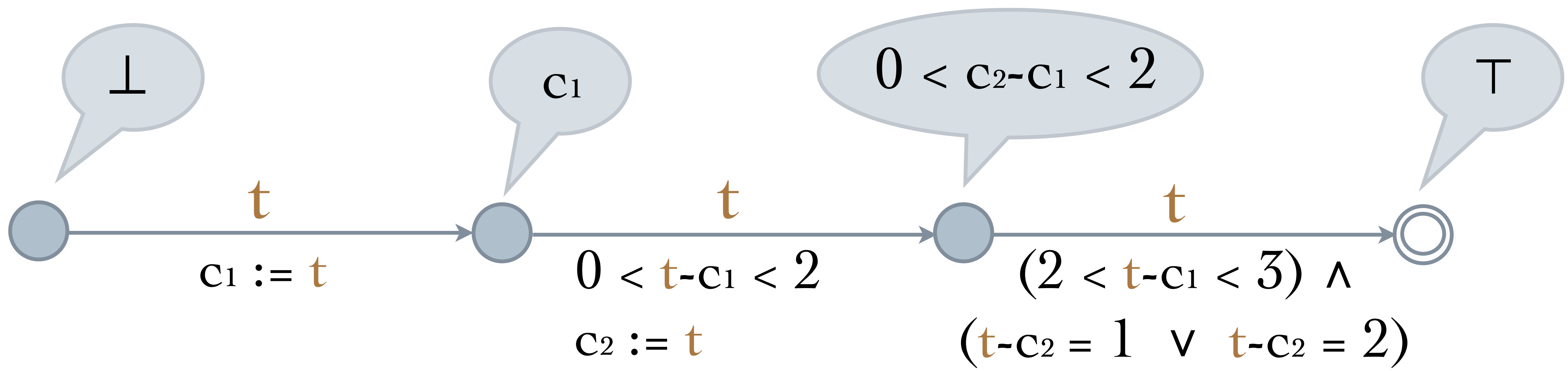
Register automata = definable NFA



states: $Q = \{\bot\} \cup R \cup \{ (c_1, c_2) \in R \times R : 0 < c_2 - c_1 < 2 \} \cup \{\top\}$

$\phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \vee c_0 + 1 = c_1 = c_2 \vee c_0 + 2 = c_1 < c_2 < c_1 + 2 \vee c_0 + 3 = c_1 = c_2$

Register automata = definable NFA

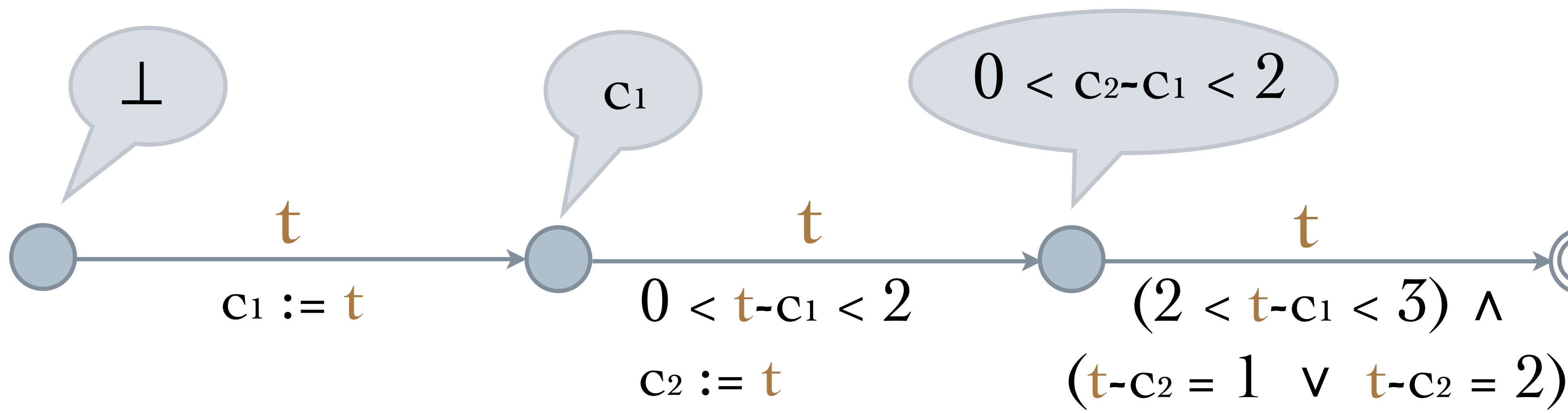


states: $Q = \{\bot\} \cup R \cup \{ (c_1, c_2) \in R \times R : 0 < c_2 - c_1 < 2 \} \cup \{\top\}$

$\phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \vee c_0 + 1 = c_1 = c_2 \vee c_0 + 2 = c_1 < c_2 < c_1 + 2 \vee c_0 + 3 = c_1 = c_2$

transitions: $\delta = \{ (\bot, t, c_1') : c_1' = t \} \cup$

Register automata = definable NFA



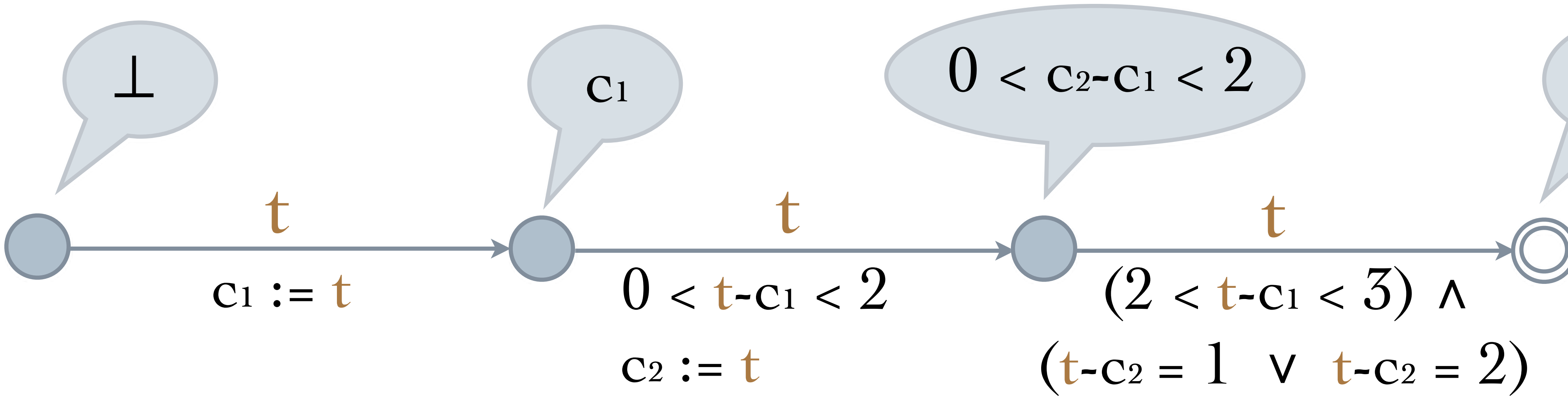
states: $Q = \{\perp\} \cup R \cup \{ (c_1, c_2) \in R \times R : 0 < c_2 - c_1 < 2 \} \cup \{\top\}$

$\phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \vee c_0+1 = c_1 = c_2 \vee c_0+2 = c_1 < c_2 < c_1+2 \vee c_0+3 = c_1 = c_2$

transitions: $\delta = \{ (\perp, t, c_1') : c_1' = t \} \cup$

$\{ (c_1, t, (c_1', c_2')) : 0 < t - c_1 < 2 \wedge c_1 = c_1' \wedge c_2' = t \} \cup$

Register automata = definable NFA



states: $Q = \{\perp\} \;\cup\; R \;\cup\; \{ (c_1, c_2) \in R \times R : 0 < c_2 - c_1 < 2 \} \;\cup\; \{\top\}$

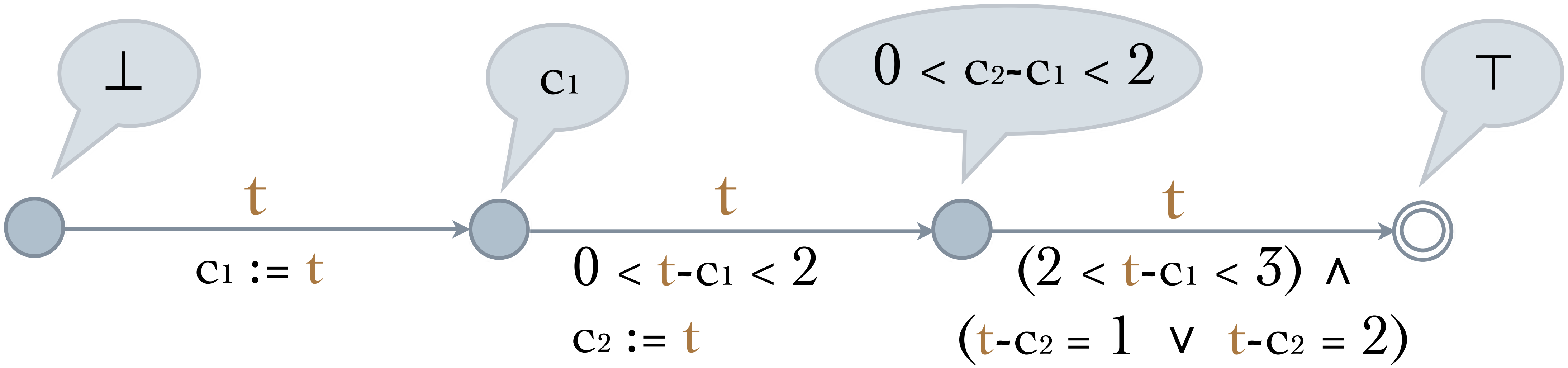
$\phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \vee c_0 + 1 = c_1 = c_2 \vee c_0 + 2 = c_1 < c_2 < c_1 + 2 \vee c_0 + 3 = c_1 = c_2$

transitions: $\delta = \{ (\perp, t, c_1') : c_1' = t \} \;\cup$

$\{ (c_1, t, (c_1', c_2')) : 0 < t - c_1 < 2 \wedge c_1 = c_1' \wedge c_2' = t \} \;\cup$

$\{ ((c_1, c_2), t, \top) : (2 < t - c_1 < 3) \wedge (t - c_2 = 1 \vee t - c_2 = 2) \}$

Register automata = definable NFA



states: $Q = \{\perp\} \cup R \cup \{ (c_1, c_2) \in R \times R : 0 < c_2 - c_1 < 2 \} \cup \{\top\}$

$\phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \vee c_0+1 = c_1 = c_2 \vee c_0+2 = c_1 < c_2 < c_1+2 \vee c_0+3 = c_1 = c_2$

transitions: $\delta = \{ (\perp, t, c_1') : c_1' = t \} \cup$
$\{ (c_1, t, (c_1', c_2')) : 0 < t - c_1 < 2 \wedge c_1 = c_1' \wedge c_2' = t \} \cup$
$\{ ((c_1, c_2), t, \top) : (2 < t - c_1 < 3) \wedge (t - c_2 = 1 \vee t - c_2 = 2) \}$

$\phi_\delta(c_0, c_1, c_2, t, c_0', c_1', c_2') \equiv \ldots$

Timed automata vs. definable NFA

Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

Timed automata vs. definable NFA

Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

- in every location, clock valuations are restricted by an orbit-finite constraint (invariant)

Timed automata vs. definable NFA

Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

- in every location, clock valuations are restricted by an orbit-finite constraint (invariant)
- number of clocks may vary from one location to another

Timed automata vs. definable NFA

Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

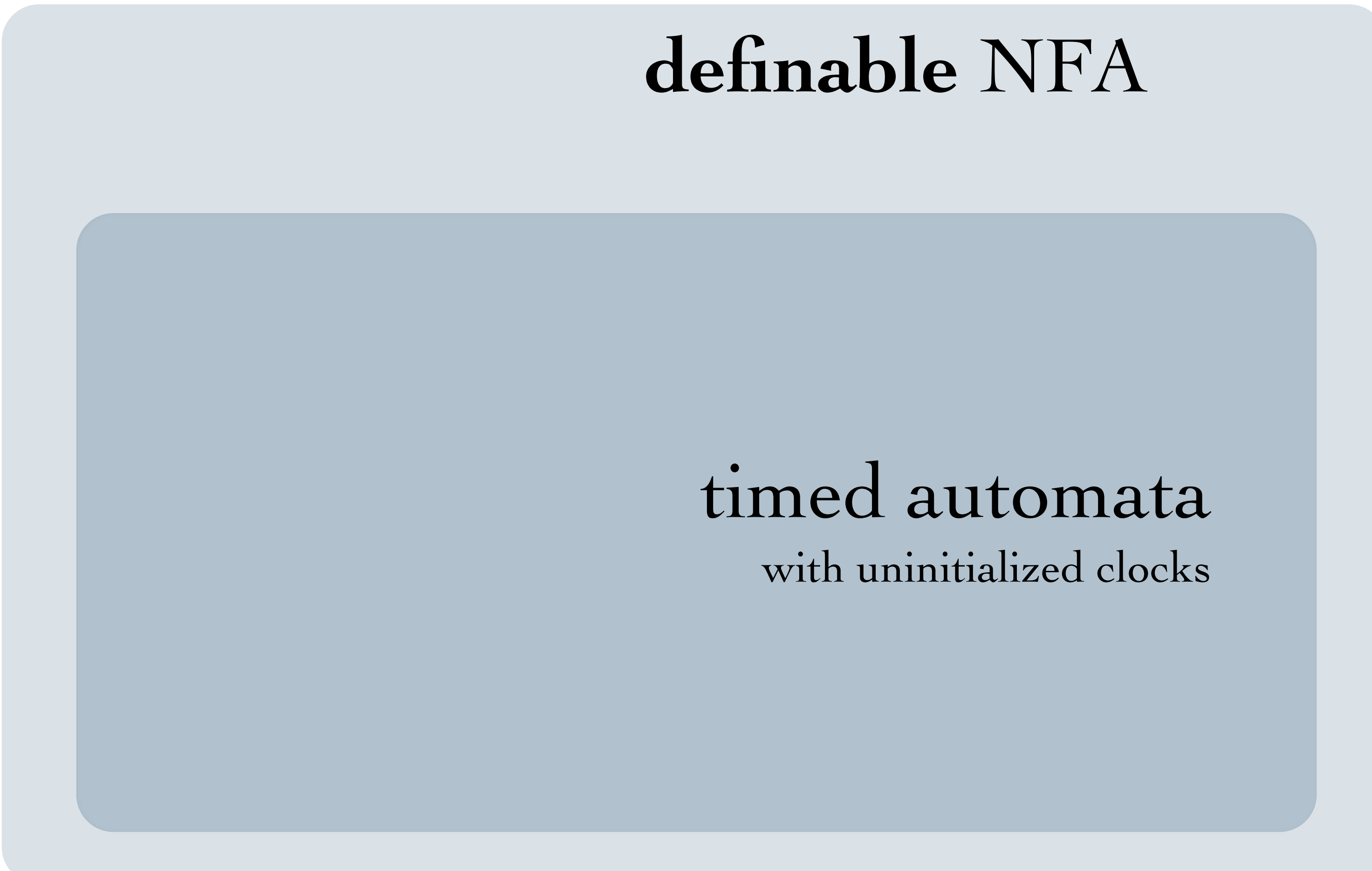
- in every location, clock valuations are restricted by an orbit-finite constraint (invariant)
- number of clocks may vary from one location to another
- the input needs not be monotonic (but can be enforced to be) nor non-negative

Timed automata vs. definable NFA

Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

- in every location, clock valuations are restricted by an orbit-finite constraint (invariant)
- number of clocks may vary from one location to another
- the input needs not be monotonic (but can be enforced to be) nor non-negative
- alphabet letters may be tuples of timestamps

Timed automata vs. definable NFA



Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata
with uninitialized clocks

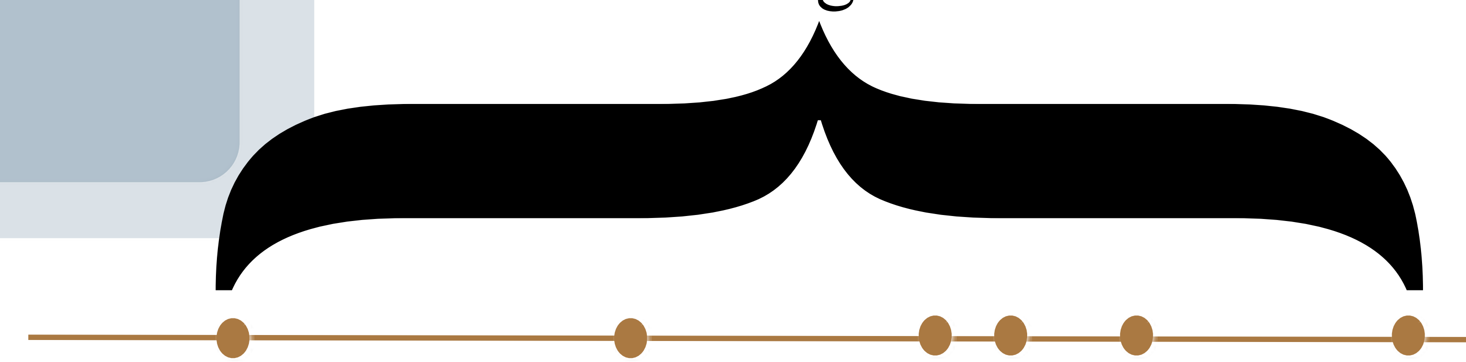
Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata

with uninitialized clocks

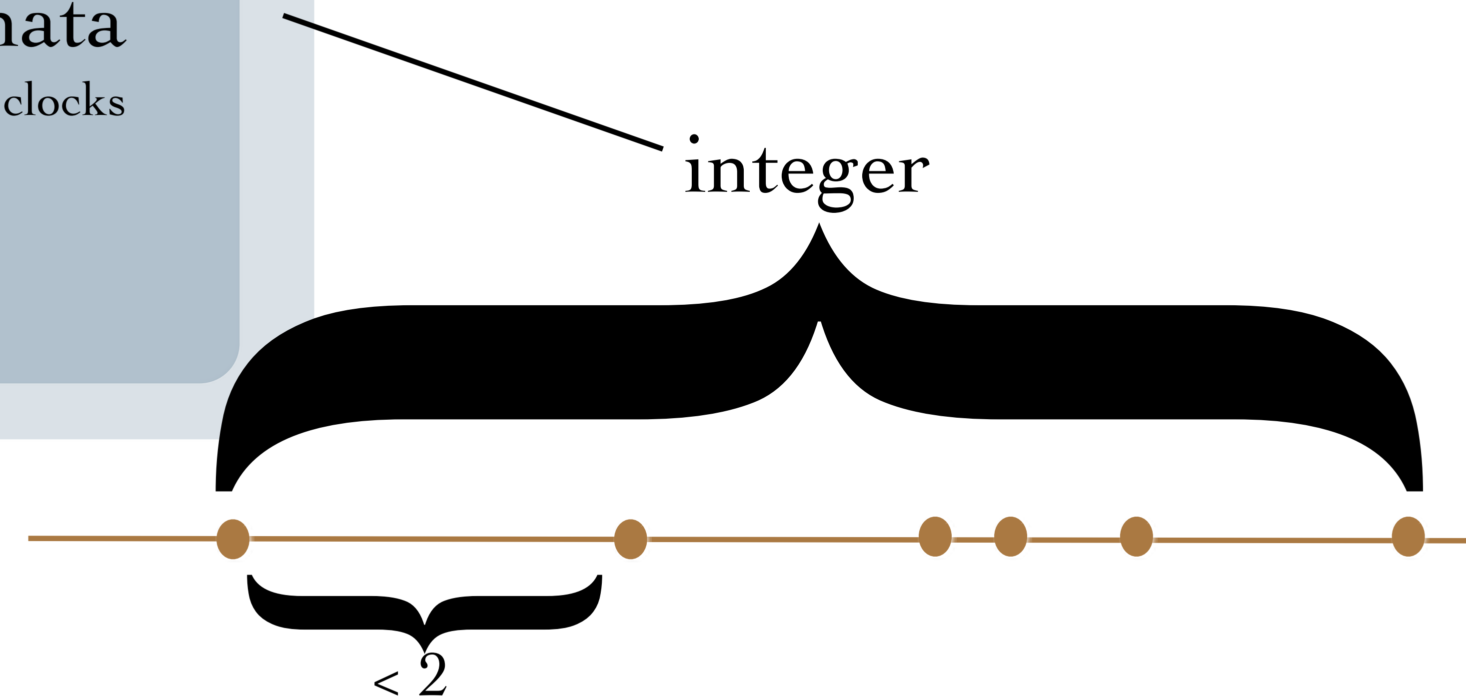
integer



Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata
with uninitialized clocks

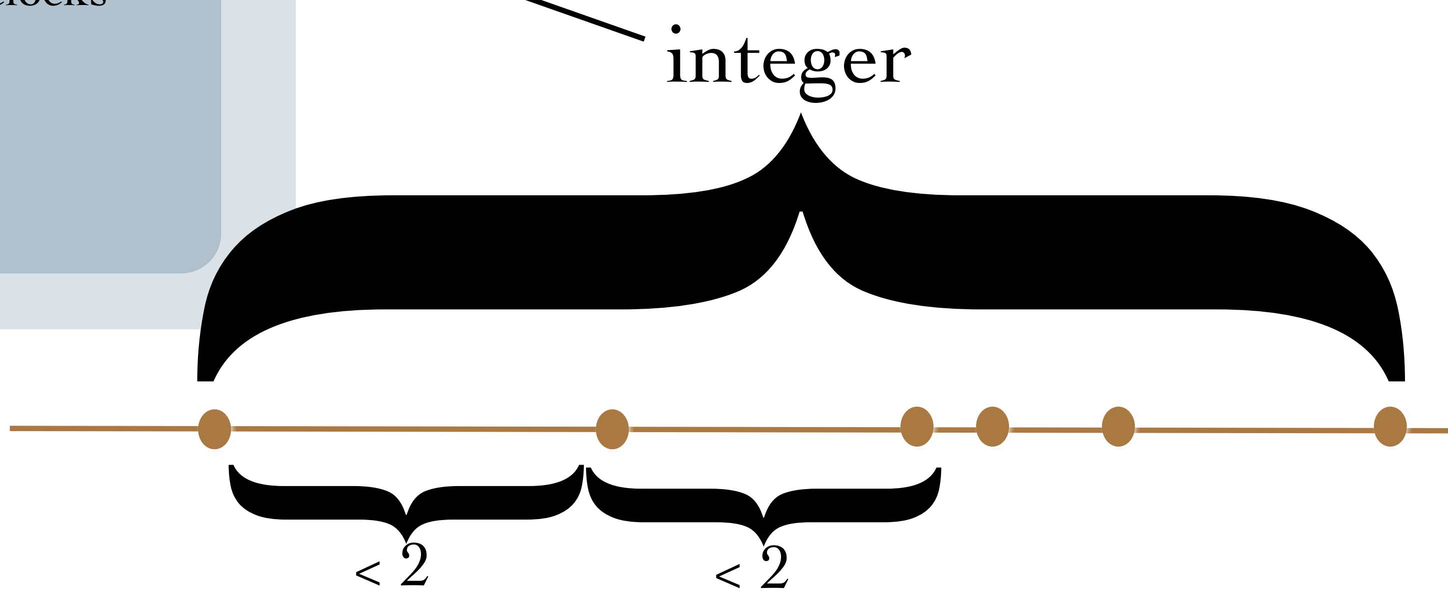


Timed automata vs. definable NFA

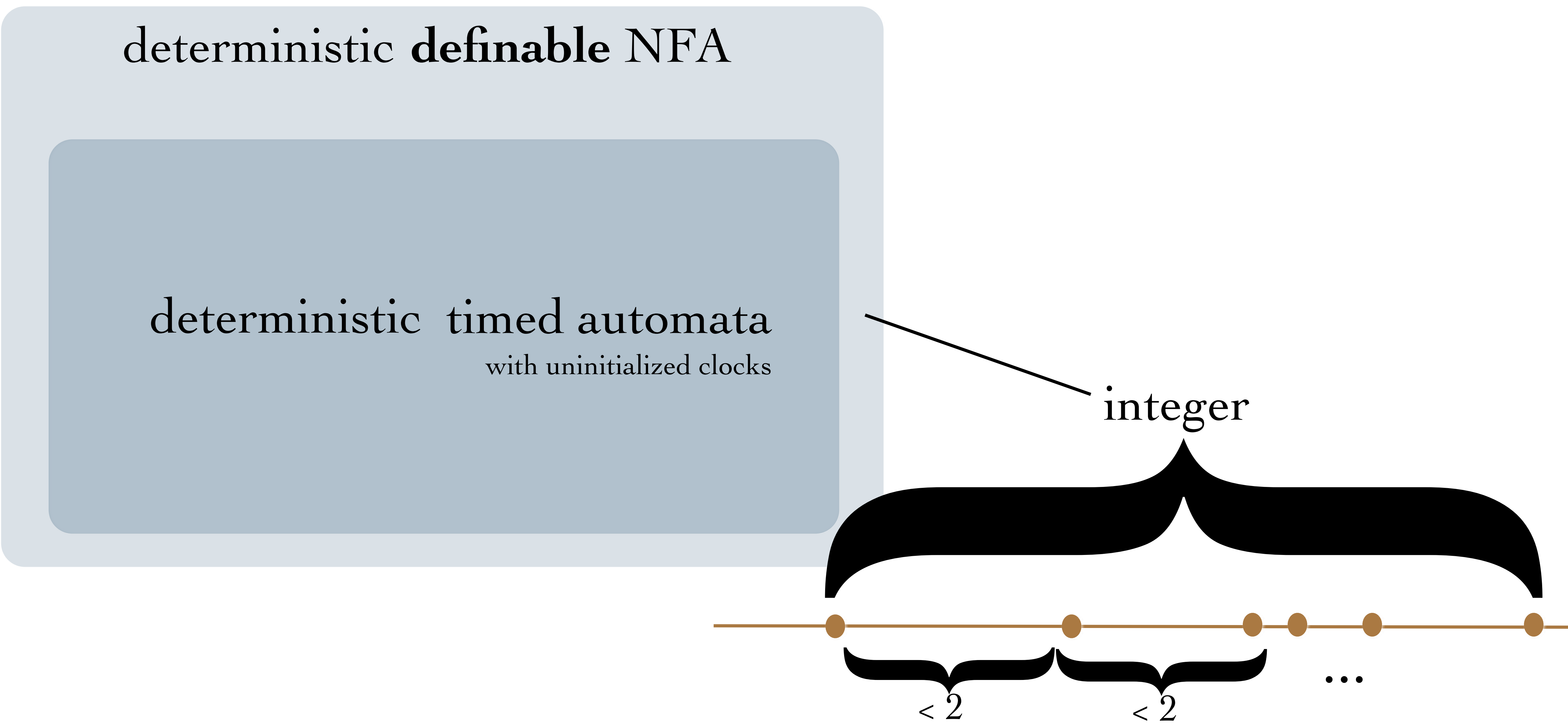
deterministic **definable** NFA

deterministic timed automata

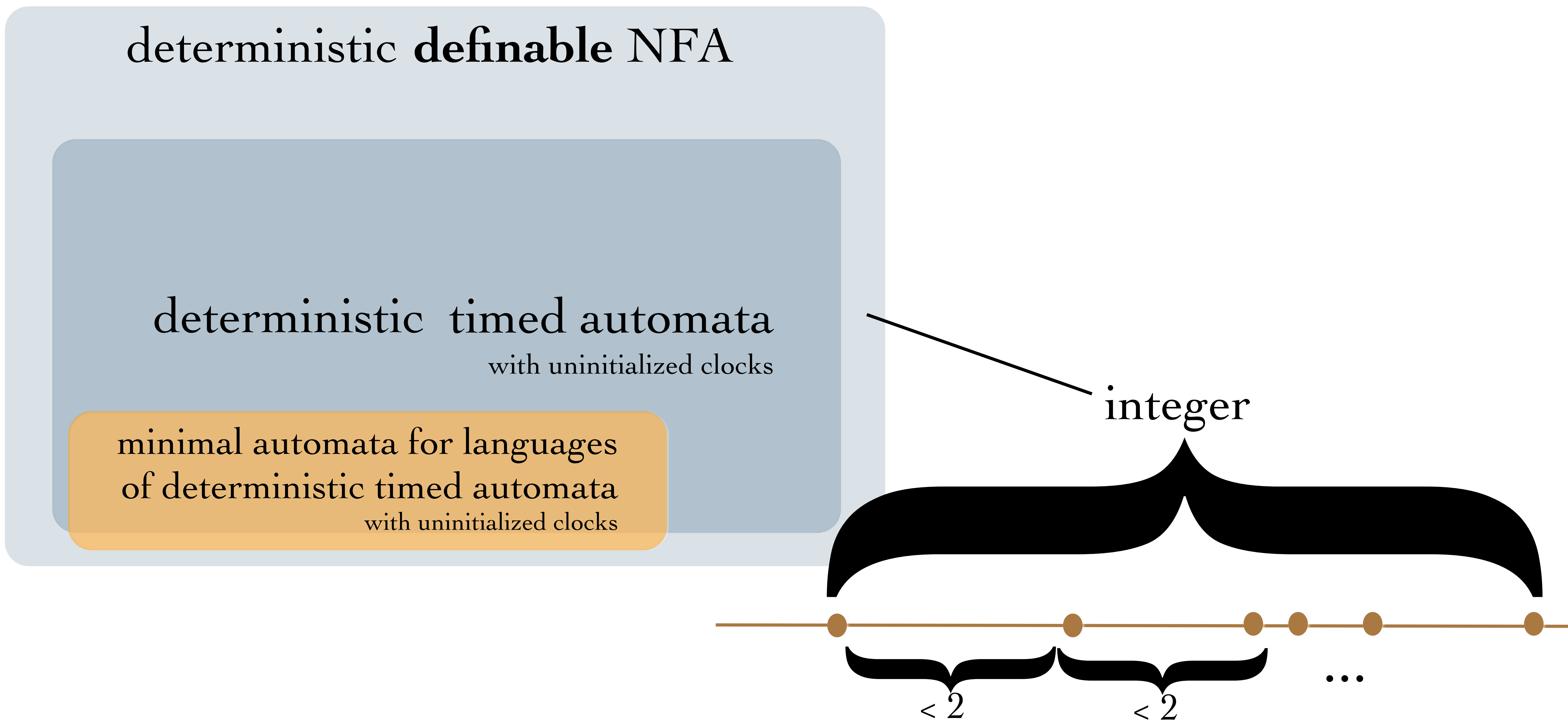
with uninitialized clocks



Timed automata vs. definable NFA



Timed automata vs. definable NFA



Timed automata vs. definable NFA

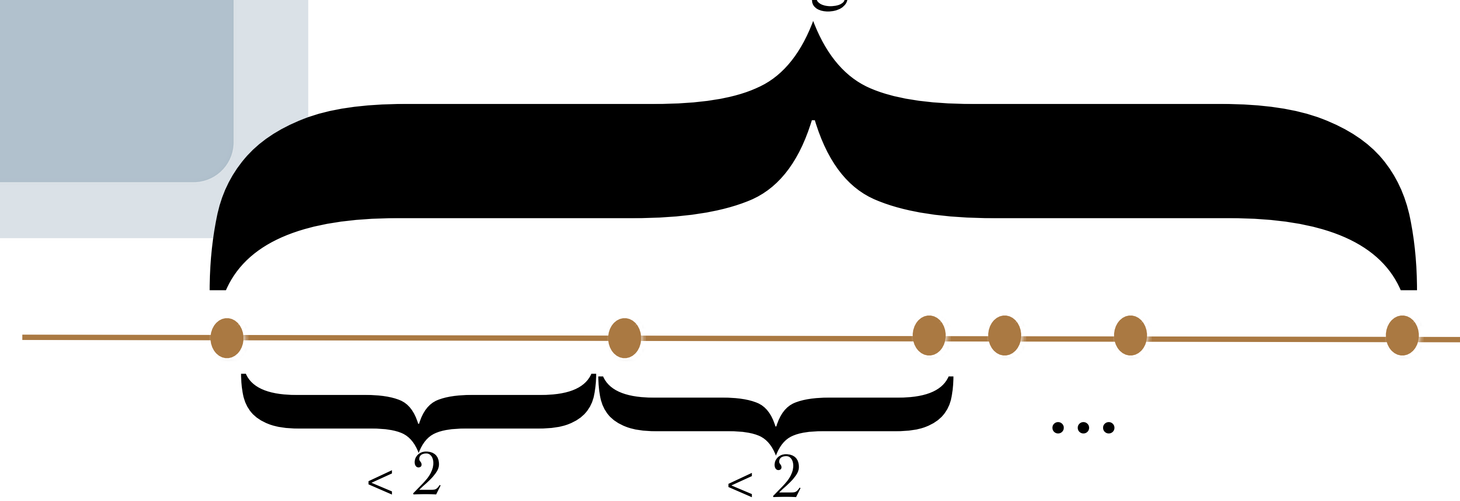
deterministic **definable** NFA

deterministic timed automata
with uninitialized clocks

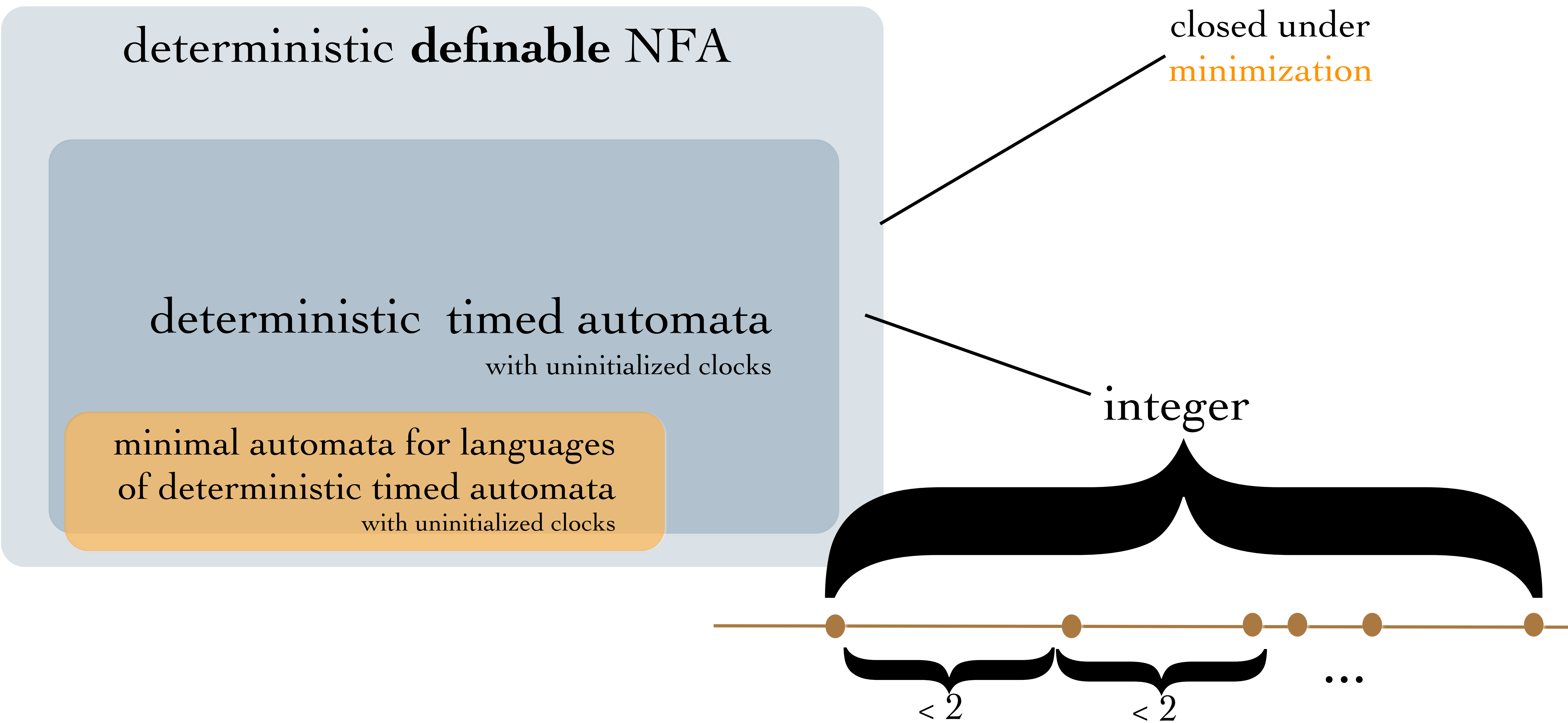
minimal automata for languages of deterministic timed automata
with uninitialized clocks

closed under minimization

integer



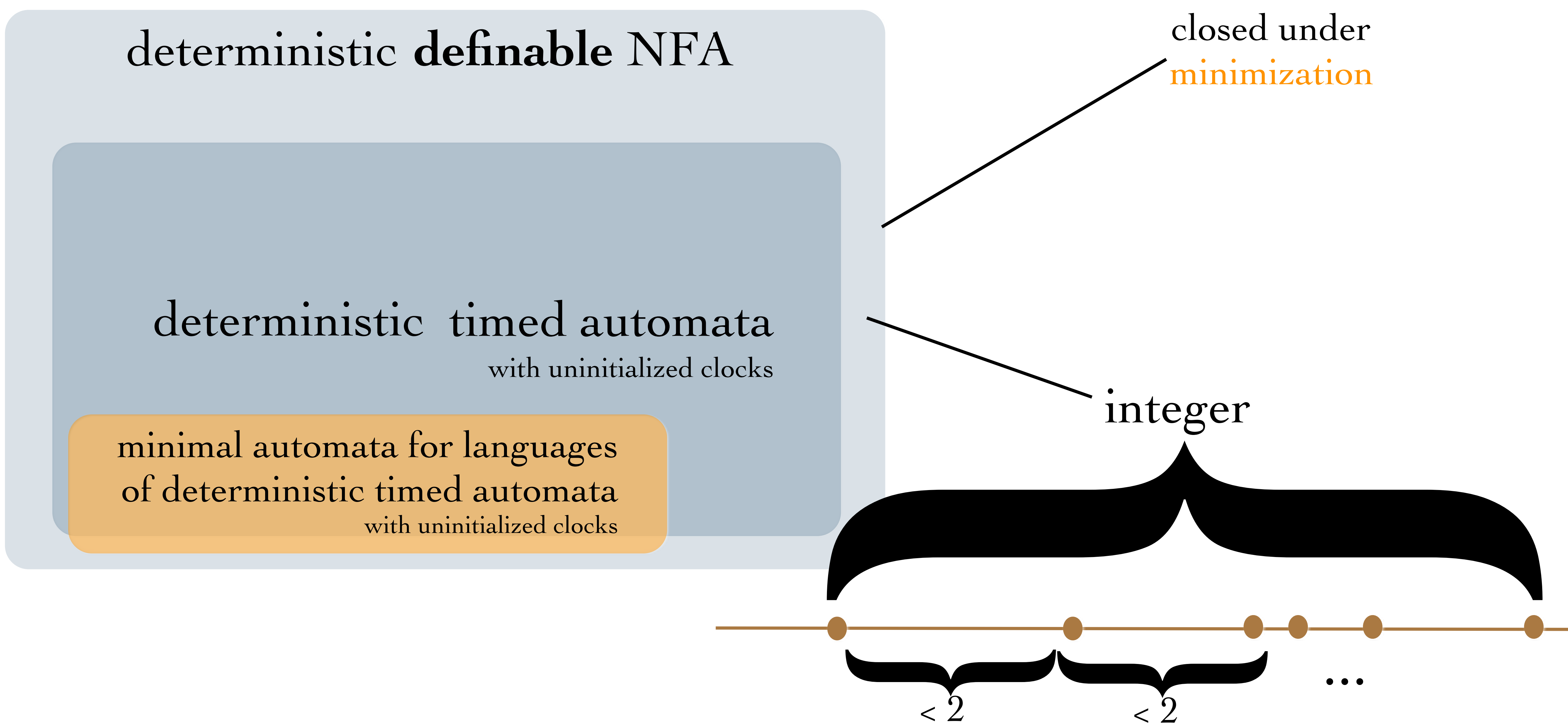
Timed automata vs. definable NFA



Theorem: [Bojańczyk, L. 2012]

Deterministic definable NFA do minimize.

Timed automata vs. definable NFA



Theorem: [Bojańczyk, L. 2012]

Deterministic definable NFA do minimize.

Likewise, if FO(<, +1) is replaced by FO(<, +).

In search of lost definition

- Motivation
- Definable NFA
- **Definable PDA**
- The core problem: equations over sets of integers
- Branching vector addition systems in dimension 1

In search of lost definition

- Motivation
- Definable NFA
- **Definable PDA**

PDA re-interpreted in definable sets

- The core problem: equations over sets of integers
- Branching vector addition systems in dimension 1

Definable PDA

- alphabet A
- states Q
- stack alphabet S
- push $\subseteq Q \times A \times Q \times S$
- pop $\subseteq Q \times S \times A \times Q$
- I, F $\subseteq Q$

orbit-finite

(<, +1)definable

Definable PDA

- alphabet A — $\phi_A(x_1, \ldots, x_n)$
- states Q — $\phi_Q(x_1, \ldots, x_m)$
- stack alphabet S — $\phi_S(x_1, \ldots, x_k)$

orbit-finite

- push $\subseteq$ Q × A × Q × S — $\phi_{\text{push}}(x_1, \ldots, x_{m+n+m+k})$
- pop $\subseteq$ Q × S × A × Q — $\phi_{\text{pop}}(x_1, \ldots, x_{m+k+n+m})$
- I, F $\subseteq$ Q — $\phi_I(x_1, \ldots, x_m),\ \phi_F(x_1, \ldots, x_m)$

Definable PDA

- alphabet A
- states Q
- stack alphabet S
- push ⊆ Q × A × Q × S
- pop ⊆ Q × S × A × Q
- I, F ⊆ Q

orbit-finite

(<, +1)-definable

$\phi_A(x_1, \ldots, x_n)$
$\phi_Q(x_1, \ldots, x_m)$
$\phi_S(x_1, \ldots, x_k)$
$\phi_{\text{push}}(x_1, \ldots, x_{m+n+m+k})$
$\phi_{\text{pop}}(x_1, \ldots, x_{m+k+n+m})$
$\phi_I(x_1, \ldots, x_m),\ \phi_F(x_1, \ldots, x_m)$

Acceptance defined as for classical PDA.

Example

input alphabet: $A = R \uplus \{\varepsilon\}$

language: "ordered palindromes of even length over reals"

states:

stack alphabet:

transitions:

initial state:

accepting state:

Example

input alphabet: $A = R \uplus \{\varepsilon\}$

language: "ordered palindromes of even length over reals"

states: $Q = R \uplus \{init, finish, acc\}$

stack alphabet:

transitions:

initial state: init

accepting state: acc

Example

input alphabet: $A = R \uplus \{\varepsilon\}$

language: "ordered palindromes of even length over reals"

states: $Q = R \uplus \{init, finish, acc\}$

stack alphabet: $S = R \uplus \{\bot\}$

transitions:

initial state: init

accepting state: acc

Example

input alphabet: $A = R \uplus \{\varepsilon\}$

language: "ordered palindromes of even length over reals"

states: $Q = R \uplus \{init, finish, acc\}$

stack alphabet: $S = R \uplus \{\bot\}$

transitions: $push \subseteq Q \times A \times Q \times S$

$(init, \varepsilon, t, \bot)$	
(t, u, u, u)	$t < u$
$(t, u, finish, u)$	$t < u$

in state init, without reading input, change state to an arbitrary real t, and push $\bot$ on stack

initial state: init

accepting state: acc

Example

input alphabet: $A = \mathbb{R} \uplus \{\varepsilon\}$

language: "ordered palindromes of even length over reals"

states: $Q = \mathbb{R} \uplus \{init, finish, acc\}$

stack alphabet: $S = \mathbb{R} \uplus \{\bot\}$

transitions: $push \subseteq Q \times A \times Q \times S$

(init, ε, t, ⊥)	
(t, u, u, u)	$t < u$
(t, u, finish, u)	$t < u$

in state finish, pop a real t from stack, read the same t from input, and stay in the same state

$pop \subseteq Q \times S \times A \times Q$

(finish, t, t, finish)	
(finish, ⊥, ε, acc)	

initial state: init

accepting state: acc

Definable prefix rewriting

- alphabet A
- states Q
- stack alphabet S
- $\rho \subseteq Q \times S^* \quad \times A \times Q \times S^*$
- $I, F \subseteq Q$

orbit-finite

(<, +1)-definable

Definable prefix rewriting

- alphabet A
- states Q
- stack alphabet S

orbit-finite

- $\rho \subseteq Q \times S^{\leq n} \times A \times Q \times S^{\leq m}$
- $I, F \subseteq Q$

$(<, +1)$-definable

Definable prefix rewriting

- alphabet A
- states Q
- stack alphabet S
- $\rho \subseteq Q \times S^{\leq n} \times A \times Q \times S^{\leq m}$
- I, F $\subseteq$ Q

orbit-finite

(<, +1)-definable

Acceptance defined as for classical prefix rewriting.

Definable context-free grammars

- nonterminal symbols S
- terminal symbols A

orbit-finite

- an initial nonterminal symbol
- $\rho \subseteq S \times (S \uplus A)^*$

definable in FO(<, +1)

Definable context-free grammars

- nonterminal symbols S
- terminal symbols A

orbit-finite

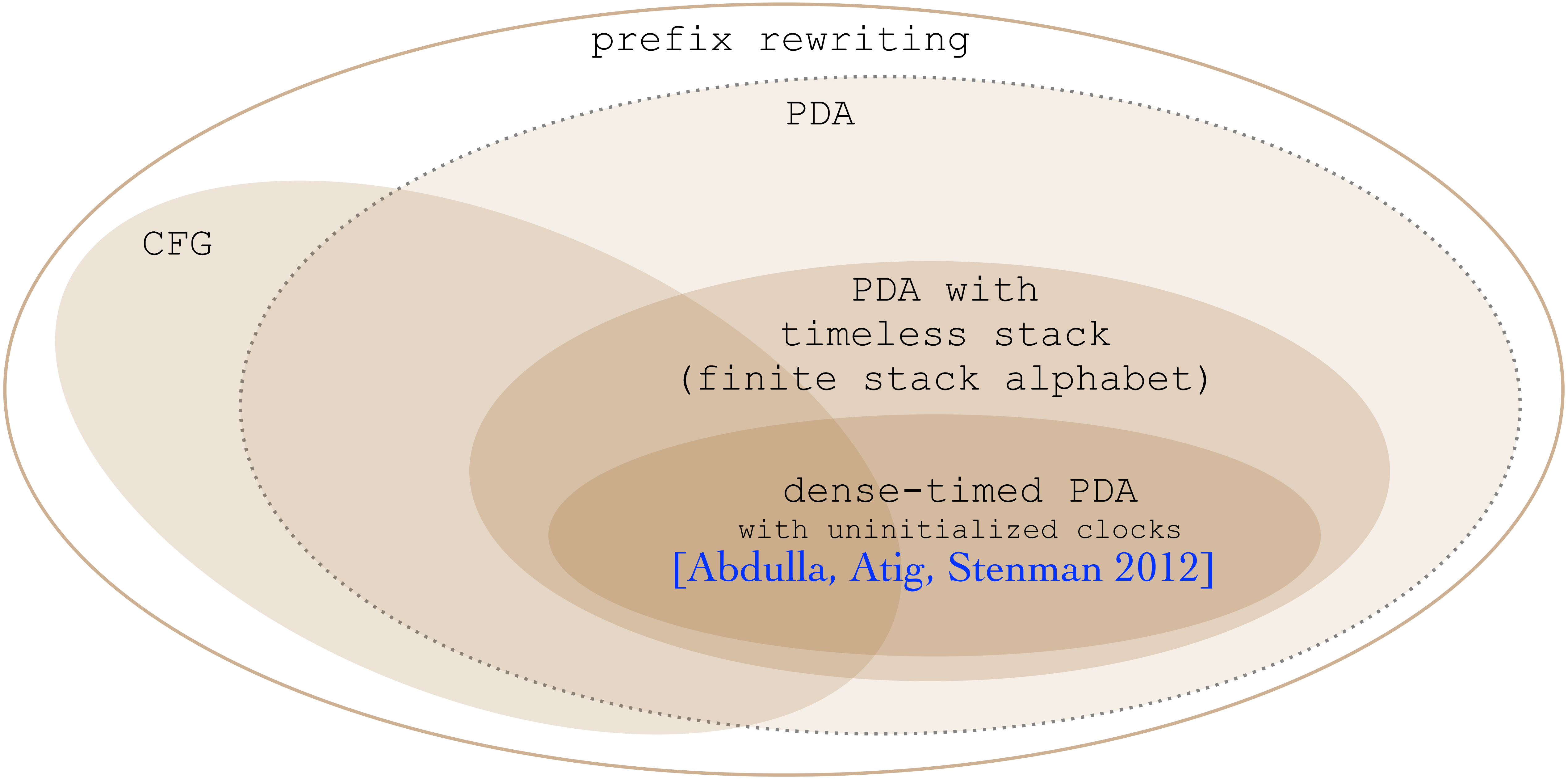
- an initial nonterminal symbol
- $\rho \subseteq S \times (S \uplus A)^{\leq n}$

definable in FO(<, +1)

Generated language defined as for classical PDA.

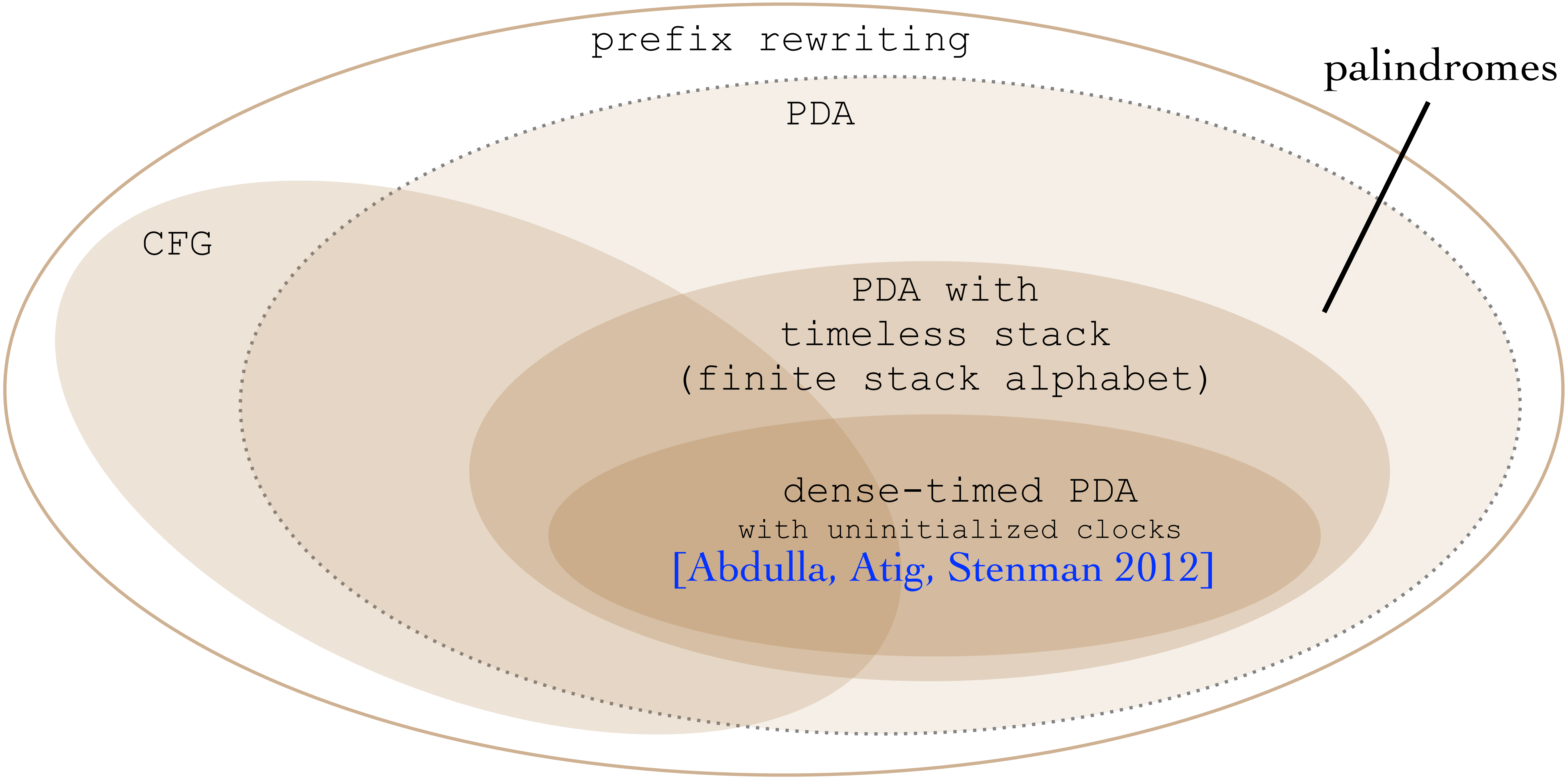
Expressiveness of definable models

[Clemente, L. 2015]



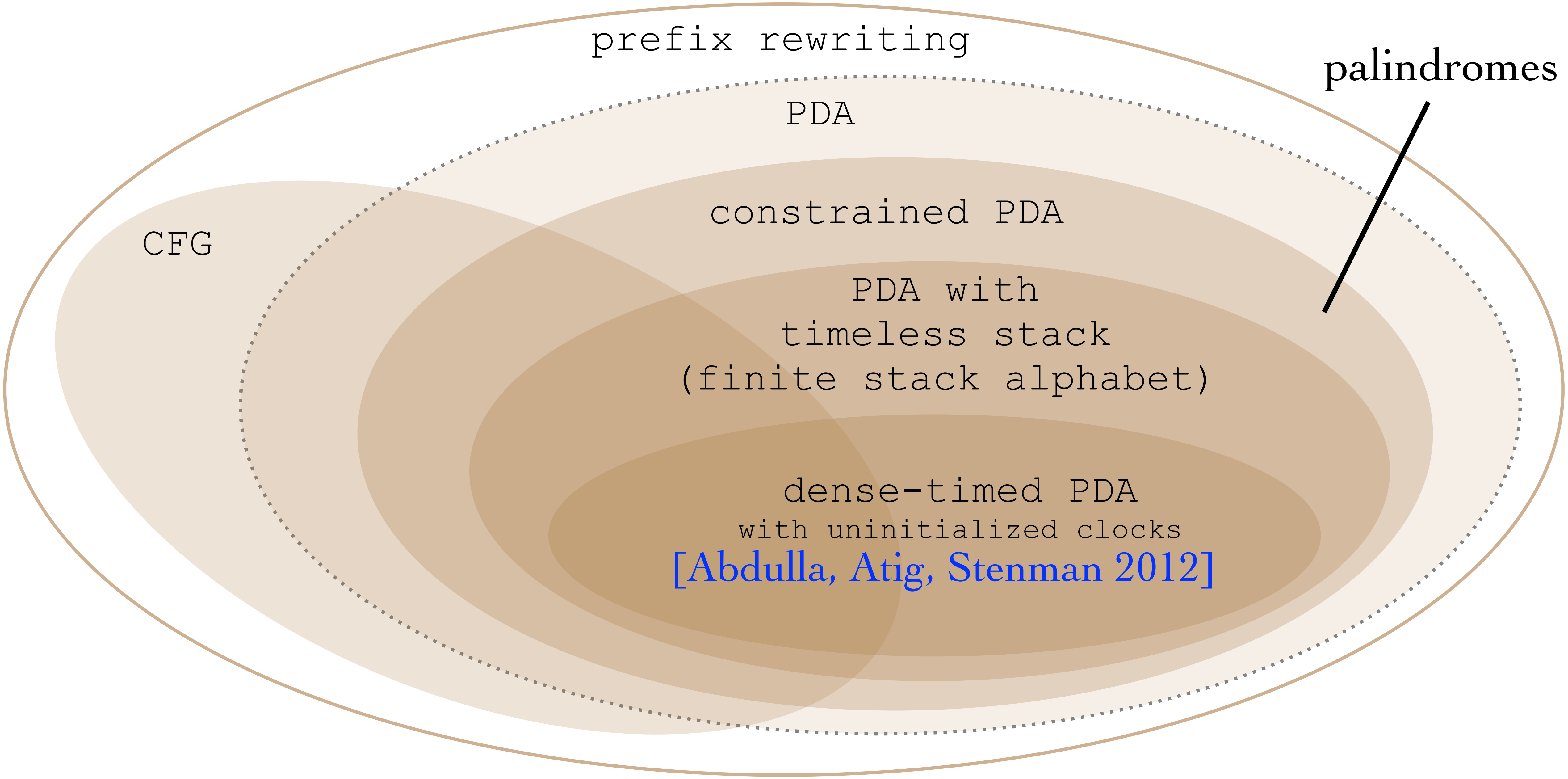
Expressiveness of definable models

[Clemente, L. 2015]



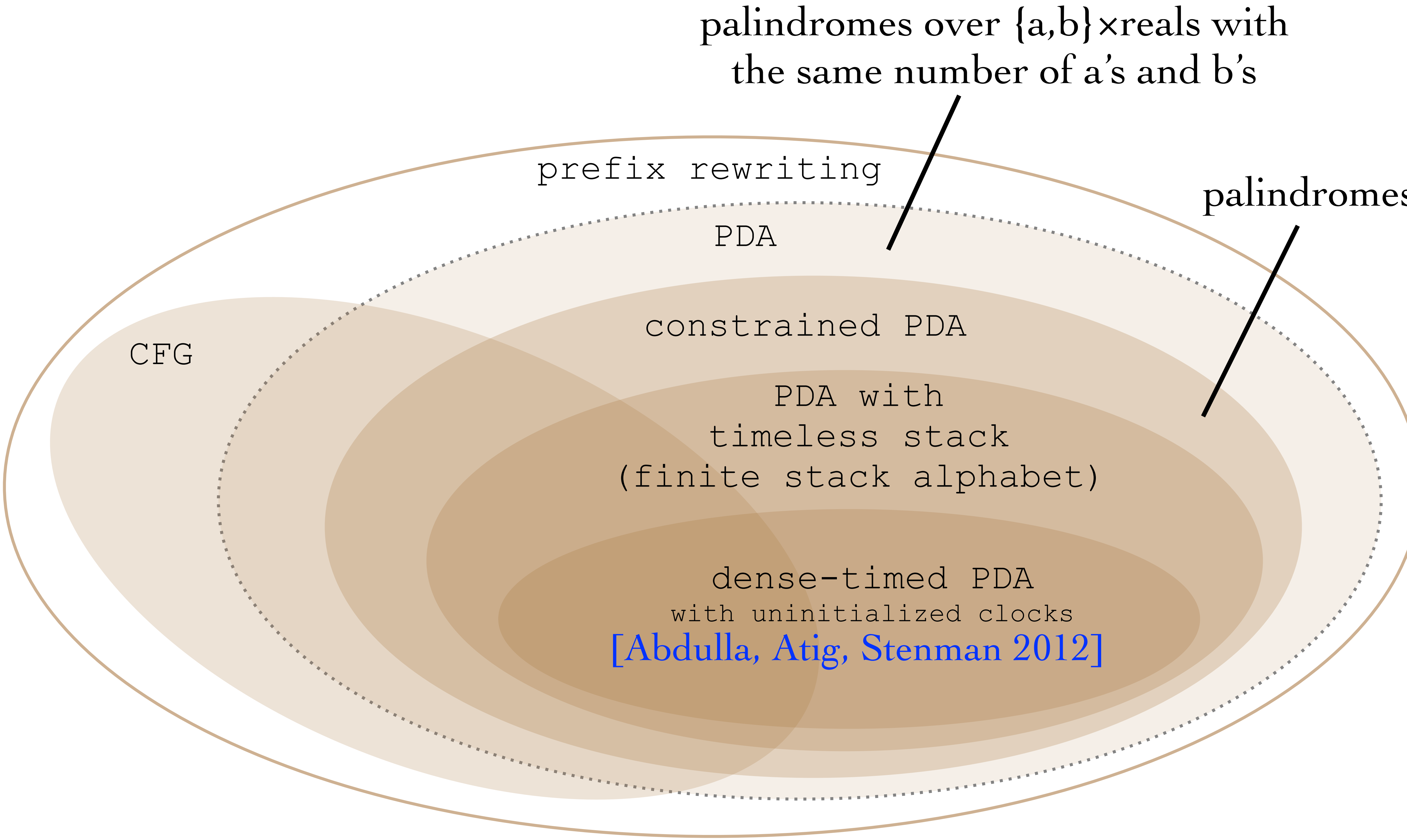
Expressiveness of definable models

[Clemente, L. 2015]



Expressiveness of definable models

[Clemente, L. 2015]



Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S
- push $\subseteq Q \times A \times Q \times S$
- pop $\subseteq Q \times S \times A \times Q$
- I, F $\subseteq Q$

orbit-finite

(<, +1)-definable

Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S

orbit-finite

- push $\subseteq$ Q $\times$ A $\times$ Q $\times$ S
- pop $\subseteq$ Q $\times$ S $\times$ A $\times$ Q
- I, F $\subseteq$ Q

orbit-finite?

Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S

orbit-finite

- push $\subseteq$ Q × A × Q × S
- pop $\subseteq$ Q × S × A × Q
- I, F $\subseteq$ Q

orbit-finite?

Span of transitions is bounded. Too strong restriction!

Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S

$\Big\}$ orbit-finite

- push $\subseteq Q \times A \times Q \times S$
- pop $\subseteq Q \times S \times A \times Q$
- I, F $\subseteq Q$

$\Big\}$ orbit-finite?

Span of transitions is bounded. Too strong restriction!

For instance, such PDA do not recognize palindromes over reals.

Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S
- push $\subseteq Q \times A \times Q \times S$
- pop $\subseteq Q \times S \times A \times Q$
- I, F $\subseteq Q$

orbit-finite

$(<, +1)$-definable

Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S
- push $\subseteq Q \times A \times \underbrace{Q \times S}_{\text{orbit-finite}}$
- pop $\subseteq \underbrace{Q \times S}_{\text{orbit-finite}} \times A \times Q$
- I, F $\subseteq$ Q

orbit-finite

(<, +1)-definable

Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S

orbit-finite

- push $\subseteq Q \times A \times \underbrace{Q \times S}_{\text{orbit-finite}}$
- pop $\subseteq \underbrace{Q \times S}_{\text{orbit-finite}} \times A \times Q$
- I, F $\subseteq$ Q

$(<, +1)$-definable

Theorem 2: [Clemente, L. 2015]
The non-emptiness problem is in NEXPTIME.
For finite stack alphabet, EXPTIME-complete.

Constrained definable PDA

- alphabet A
- states Q
- stack alphabet S

orbit-finite

- push $\subseteq Q \times A \times \underbrace{Q \times S}_{\text{orbit-finite}}$
- pop $\subseteq \underbrace{Q \times S}_{\text{orbit-finite}} \times A \times Q$
- $I, F \subseteq Q$

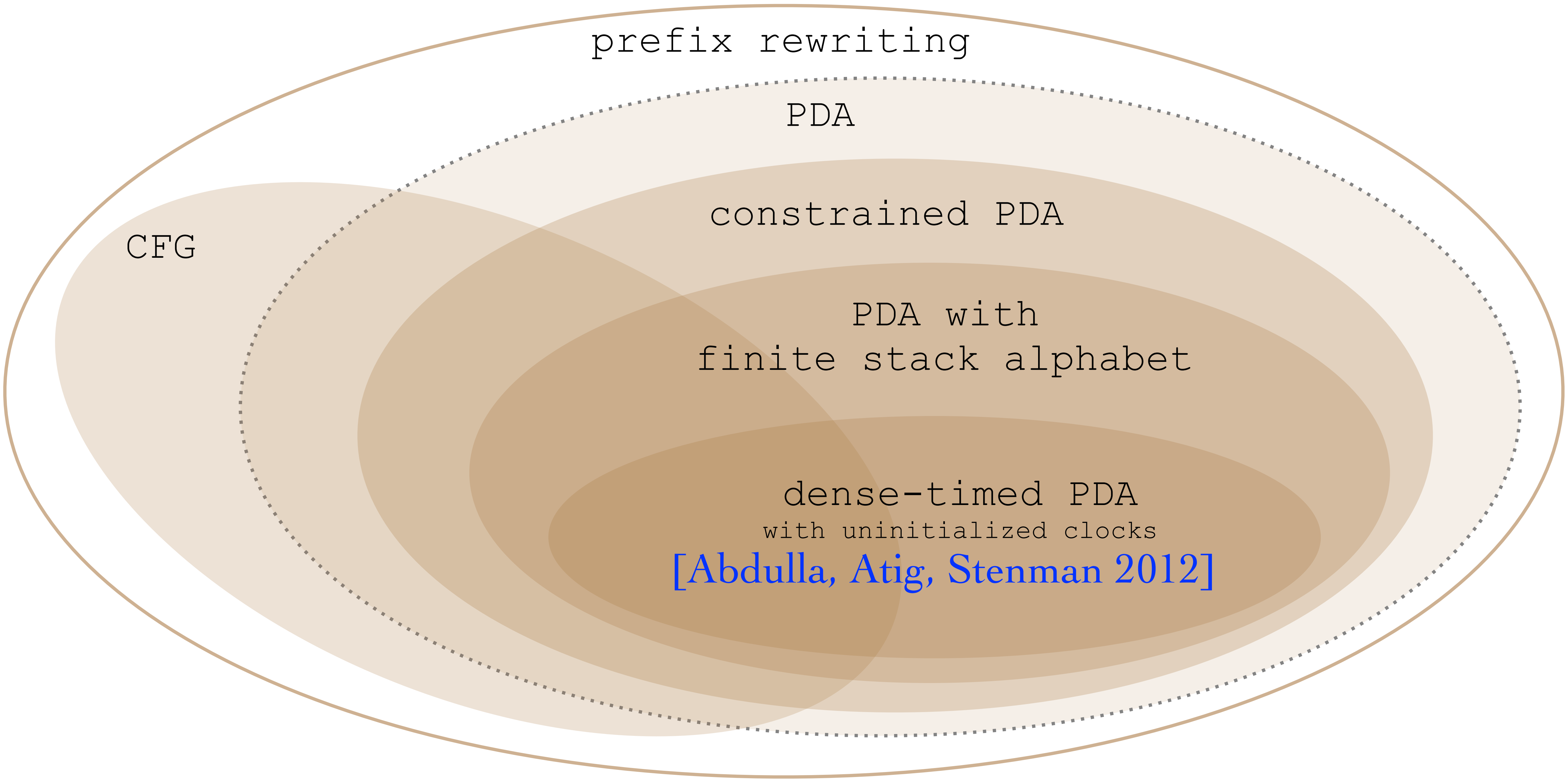
$(<, +1)$-definable

Theorem 2: [Clemente, L. 2015]
The non-emptiness problem is in NEXPTIME.
For finite stack alphabet, EXPTIME-complete.

Fact: The model subsumes dense-timed PDA with uninitialized clocks.

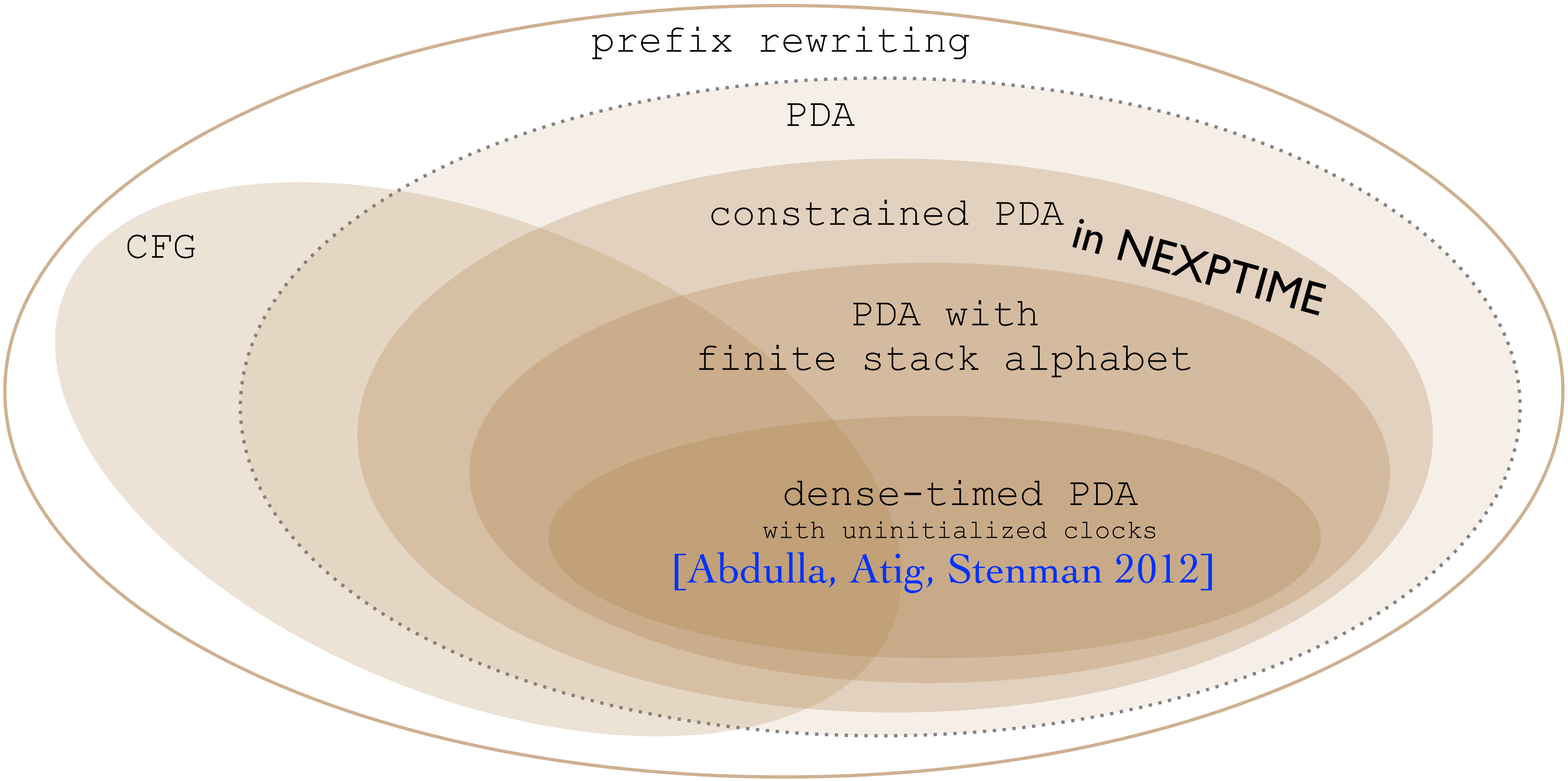
Decidability of non-emptiness

[Clemente, L. 2015]



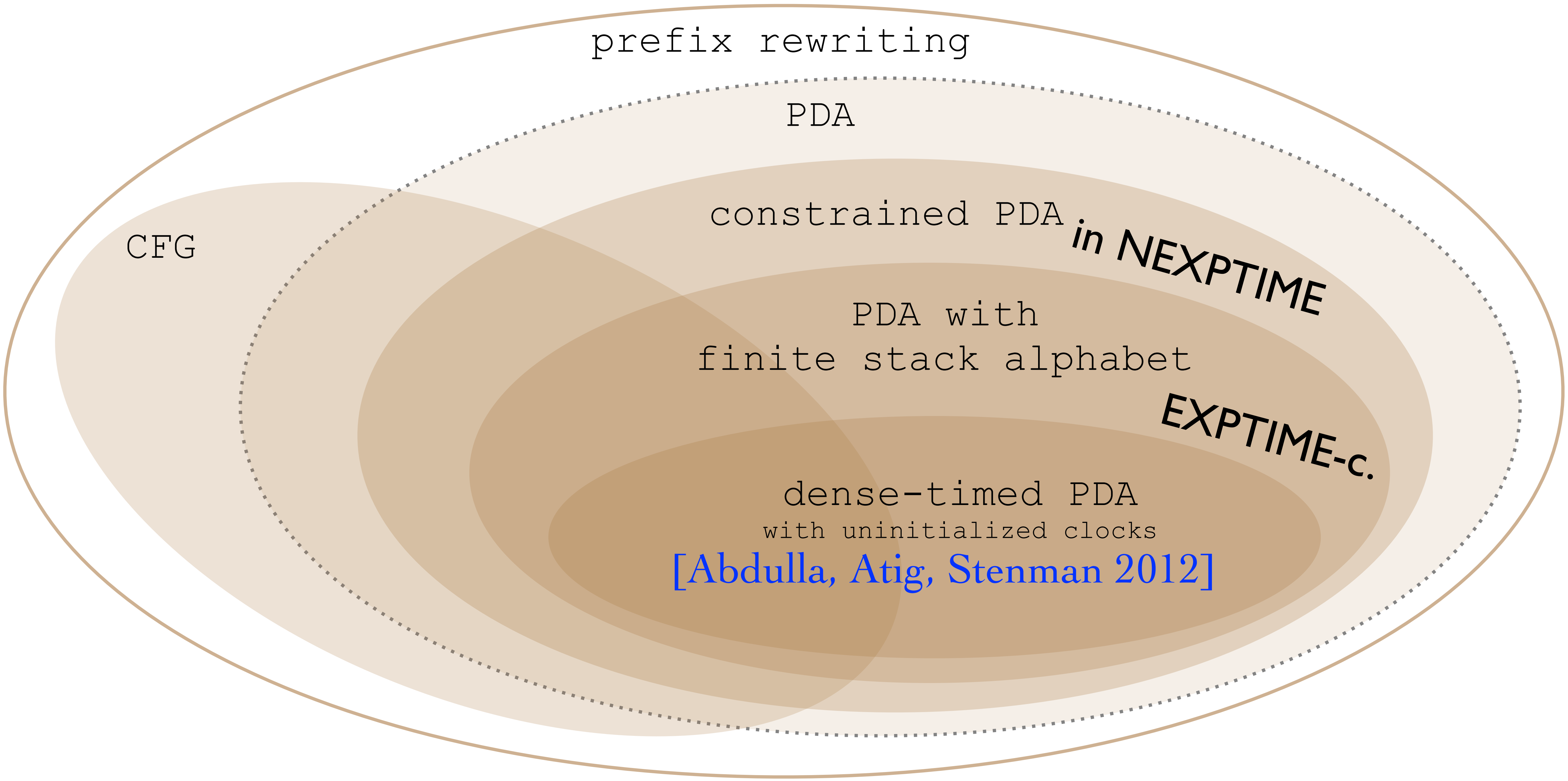
Decidability of non-emptiness

[Clemente, L. 2015]



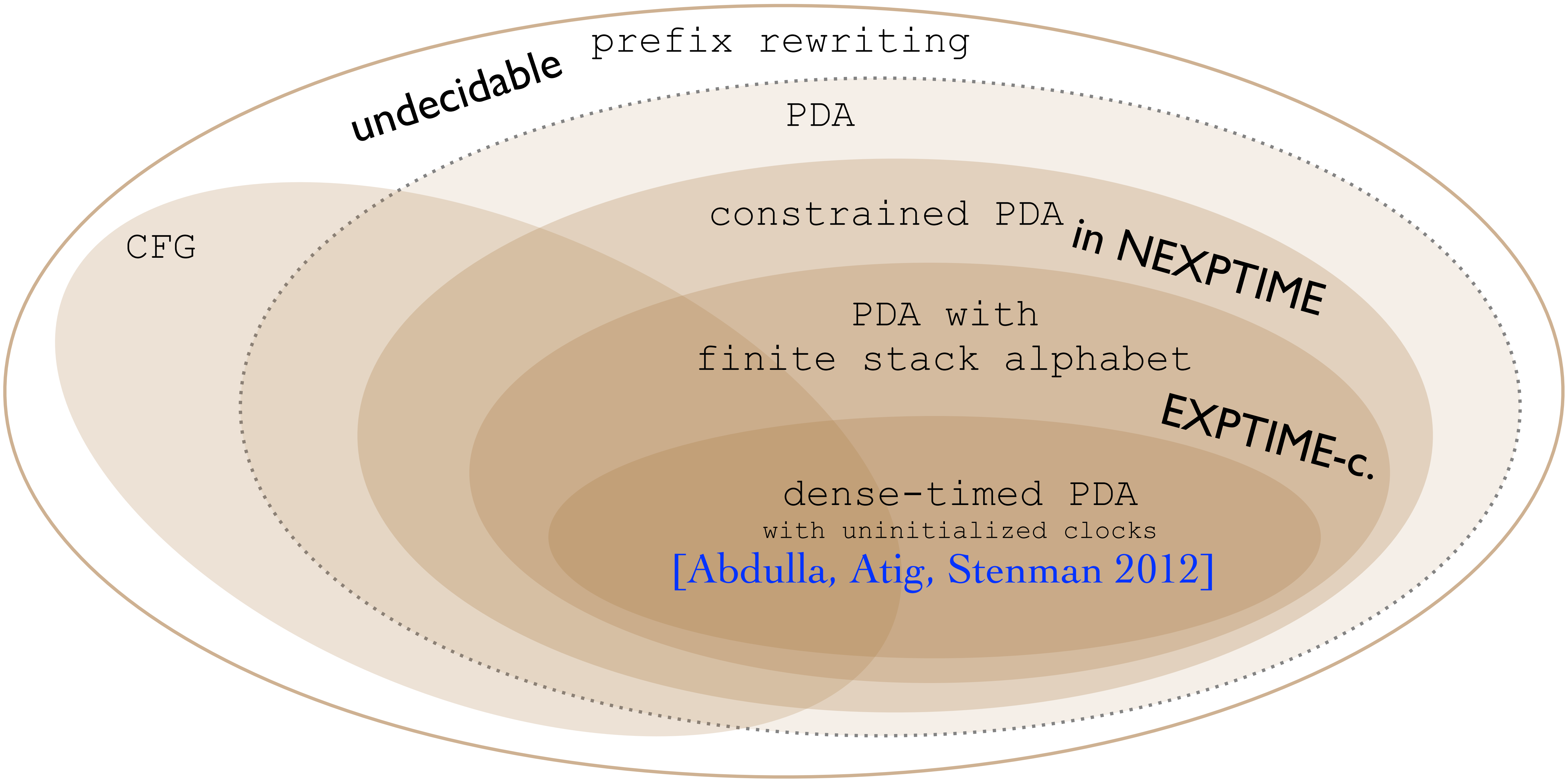
Decidability of non-emptiness

[Clemente, L. 2015]



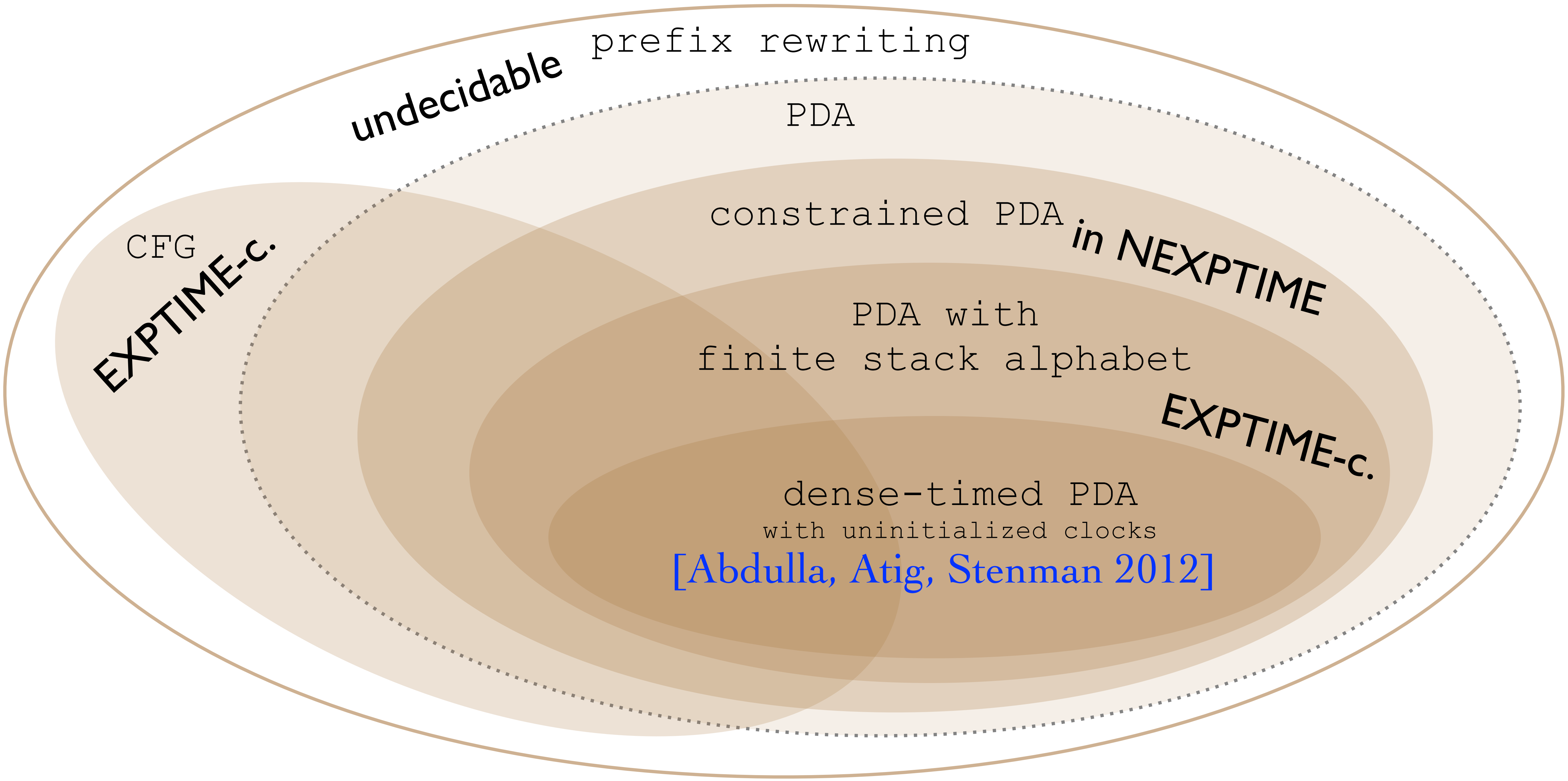
Decidability of non-emptiness

[Clemente, L. 2015]



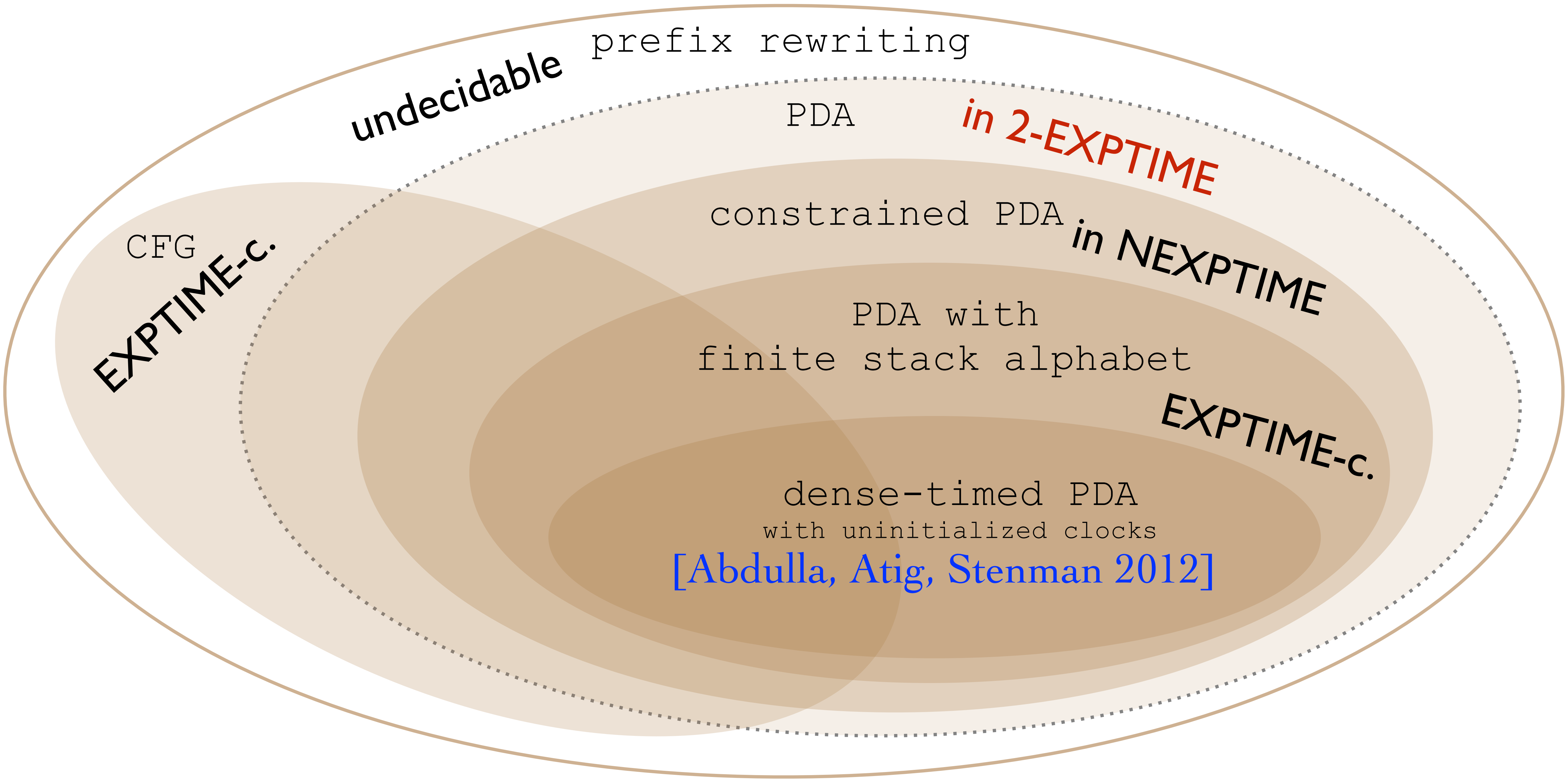
Decidability of non-emptiness

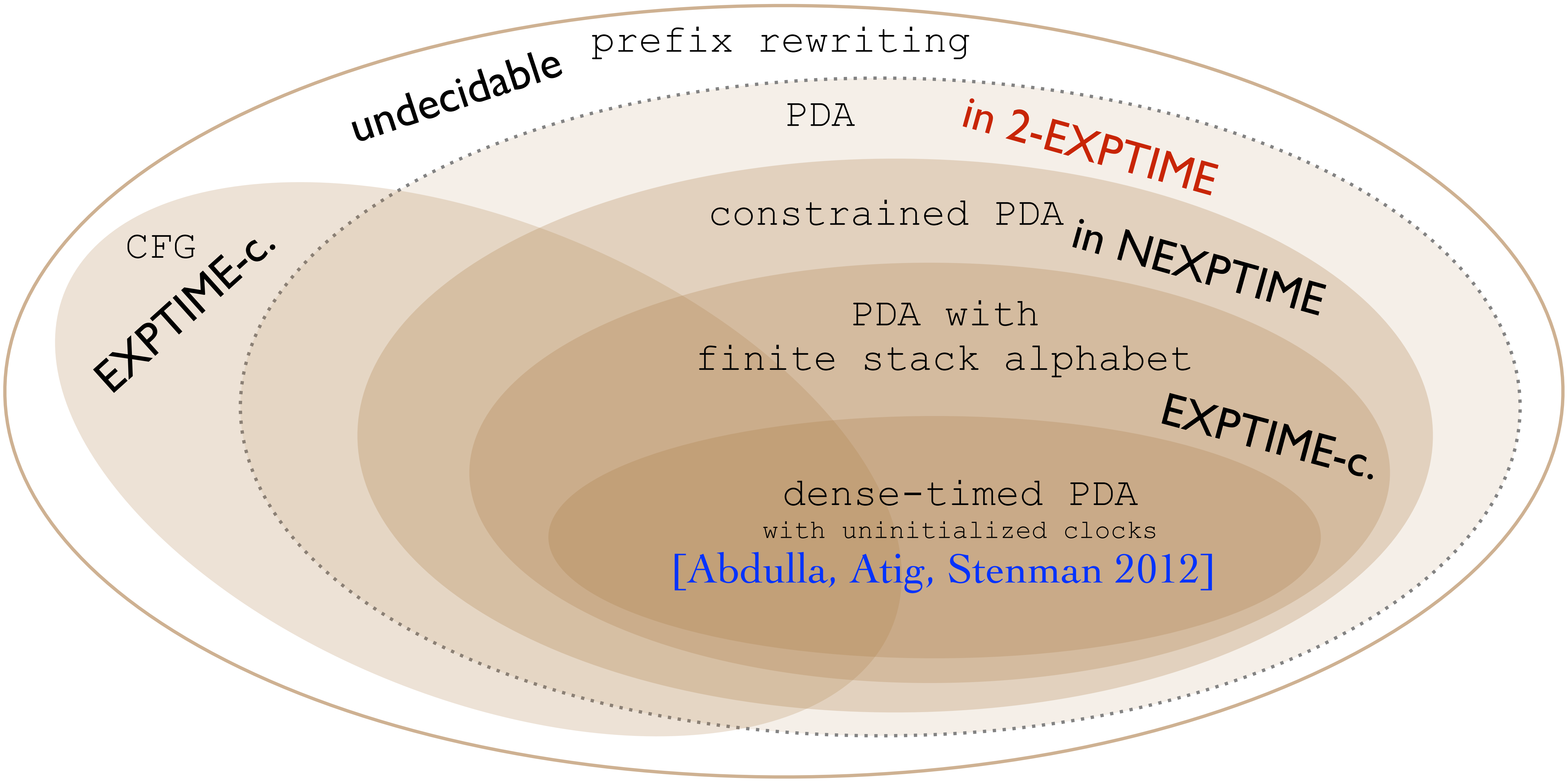
[Clemente, L. 2015]

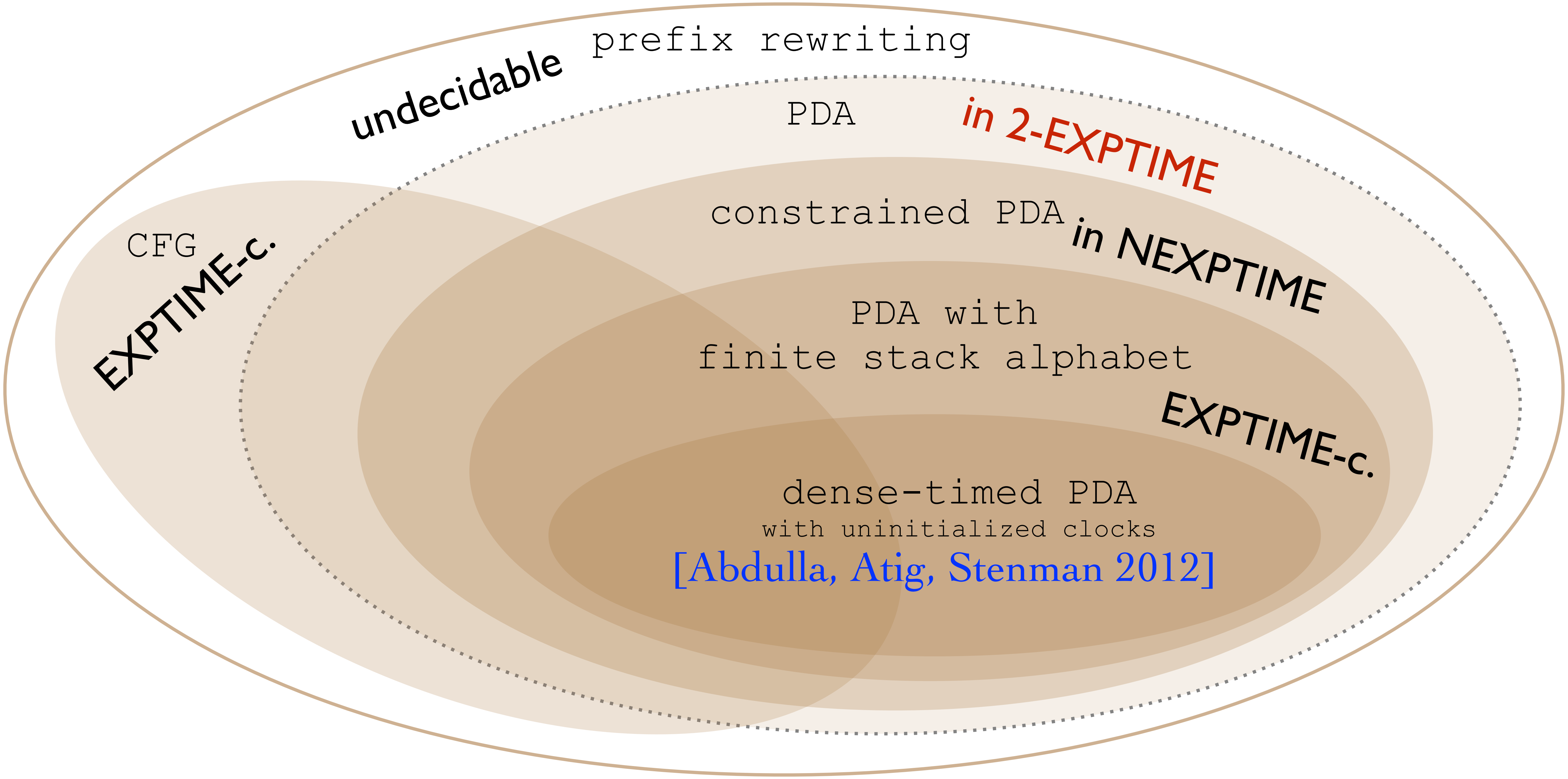


Decidability of non-emptiness

[Clemente, L. 2015]

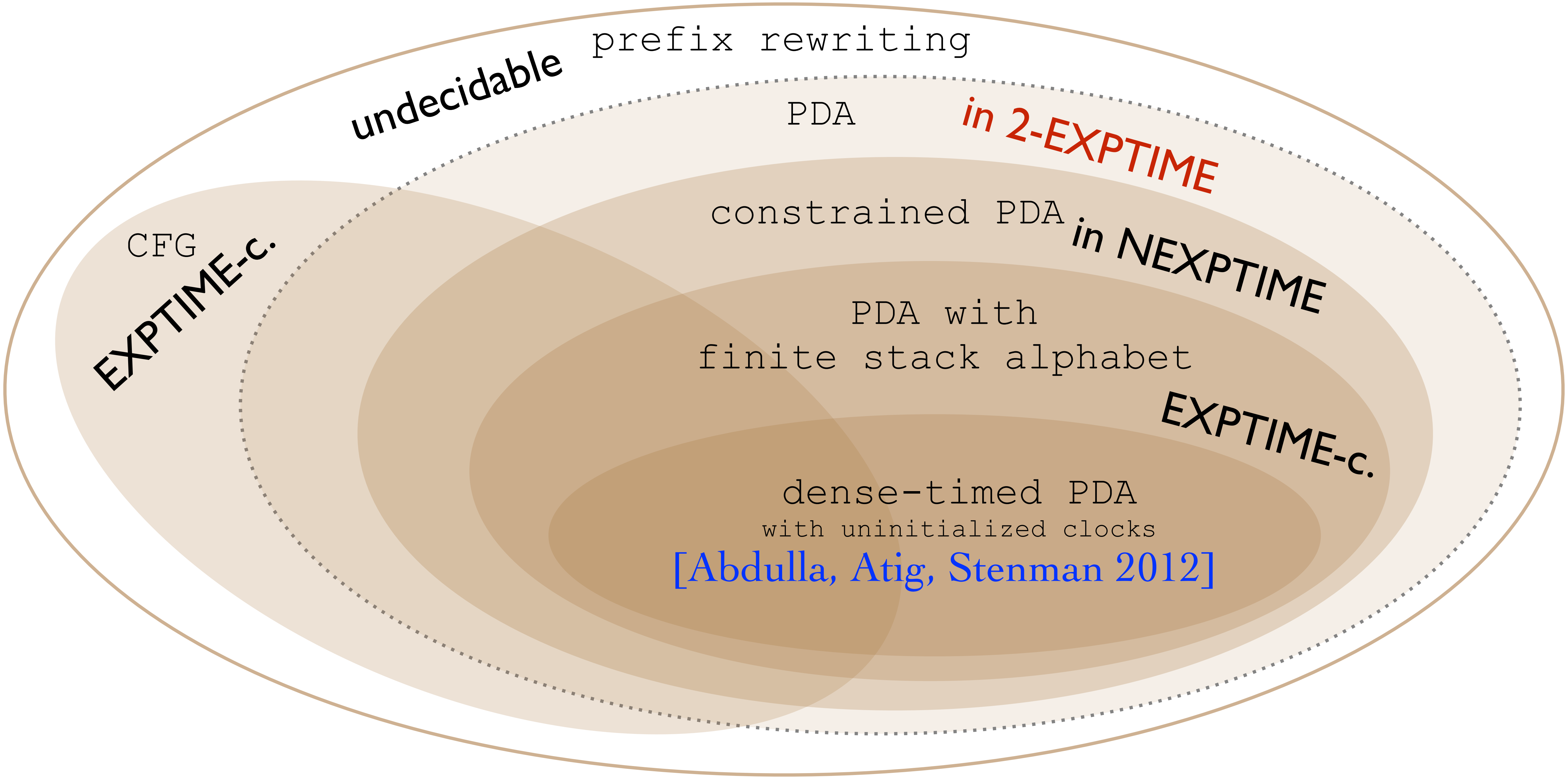



prefix rewriting
undecidable
PDA
in 2-EXPTIME
constrained PDA
in NEXPTIME
CFG
EXPTIME-c.
PDA with
finite stack alphabet
EXPTIME-c.
dense-timed PDA
with uninitialized clocks
[Abdulla, Atig, Stenman 2012]



Theorem 3:

The non-emptiness problem of definable PDA
is in 2-EXPTIME.



Theorem 3:

The non-emptiness problem of definable PDA is in 2-EXPTIME.

Complexity gap: EXPTIME … 2-EXPTIME

Towards decision procedure

Towards decision procedure

Notation: $q \dashrightarrow p$ — there is a run from state p to state q that starts and ends with the empty stack

Towards decision procedure

Notation: $q \dashrightarrow p$ — there is a run from state p to state q that starts and ends with the empty stack

(base) $$\frac{}{x \dashrightarrow x}$$

Towards decision procedure

Notation: $q \dashrightarrow p$ — there is a run from state p to state q that starts and ends with the empty stack

(base) $$\frac{}{x \dashrightarrow x}$$

(transitivity) $$\frac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$$

Towards decision procedure

Notation: $q \dashrightarrow p$ — there is a run from state p to state q that starts and ends with the empty stack

(base) $$\frac{}{x \dashrightarrow x}$$

(transitivity) $$\frac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$$

(push-pop) $$\frac{x \dashrightarrow y}{x' \dashrightarrow y'}$$ if push(x', x, s) and pop(y, s, y') for some stack symbol s

Towards decision procedure

Notation: $q \dashrightarrow p$ — there is a run from state p to state q that starts and ends with the empty stack

(base) $$\frac{}{x \dashrightarrow x}$$

(transitivity) $$\frac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$$

(push-pop) $$\frac{x \dashrightarrow y}{x' \dashrightarrow y'}$$ if push(x', x, s) and pop(y, s, y') for some stack symbol s

Problem: how to make this work for orbit-finite state space?

Towards decision procedure

Notation: $q \dashrightarrow p$ — there is a run from state p to state q that starts and ends with the empty stack

(base) $$\frac{}{x \dashrightarrow x}$$

(transitivity) $$\frac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$$

(push-pop) $$\frac{x \dashrightarrow y}{x' \dashrightarrow y'}$$ if push(x', x, s) and pop(y, s, y') for some stack symbol s

Problem: how to make this work for orbit-finite state space?

Guideline: think like state = an integer

Towards decision procedure

Notation: $q \dashrightarrow p$ — there is a run from state p to state q that starts and ends with the empty stack

(base)
$$\frac{}{x \dashrightarrow x}$$

(transitivity)
$$\frac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$$

(push-pop)
$$\frac{x \dashrightarrow y}{x' \dashrightarrow y'}$$
if push(x', x, s) and pop(y, s, y') for some stack symbol s

Problem: how to make this work for orbit-finite state space?

Guideline: think like state = an integer

capture all differences y - x, for $x \dashrightarrow y$

Towards decision procedure

- Motivation
- Definable NFA
- Definable PDA
- **The core problem: equations over sets of integers**
- Branching vector addition systems in dimension 1

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

- constants {-1}, {0}, {1}

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

- constants {-1}, {0}, {1}
- set union $\cup$

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

decide, whether its least solution assigns a non-empty set to x_1?

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \cdots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

decide, whether its least solution assigns a non-empty set to x_1?

for instance:

$$\begin{cases} x_1 & = & \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\ x_2 & = & x_1 + \{1\} \cup x_1 + \{-1\} \end{cases}$$

The core problem: non-emptiness

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

where right-hand sides use:

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

decide, whether its least solution assigns a non-empty set to x_1?

for instance:

$$\begin{cases} x_1 & = & \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\ x_2 & = & x_1 + \{1\} \cup x_1 + \{-1\} \end{cases}$$

What is the least solution with respect to inclusion?

definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

(base) $$\frac{}{x \dashrightarrow x}$$

(transitivity) $$\frac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$$

(push-pop) $$\frac{x \dashrightarrow y}{x' \dashrightarrow y'}$$

definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

(base) $\dfrac{}{x \dashrightarrow x}$

(transitivity) $\dfrac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$

(push-pop) $\dfrac{x \dashrightarrow y}{x' \dashrightarrow y'}$

Guideline:

think like state = an integer,

capture all differences y - x,

for $x \dashrightarrow y$

definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

(base) $$\frac{}{x \dashrightarrow x}$$ $X_{pp} \supseteq \{0\}$

(transitivity) $$\frac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$$

(push-pop) $$\frac{x \dashrightarrow y}{x' \dashrightarrow y'}$$

Guideline:

think like state = an integer,

capture all differences y - x,

for $x \dashrightarrow y$

definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

(base) $\dfrac{}{x \dashrightarrow x}$ $X_{pp} \supseteq \{0\}$

(transitivity) $\dfrac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$ $X_{pr} \supseteq X_{pq} + X_{qr}$

(push-pop) $\dfrac{x \dashrightarrow y}{x' \dashrightarrow y'}$

Guideline:

think like state = an integer,

capture all differences y - x,

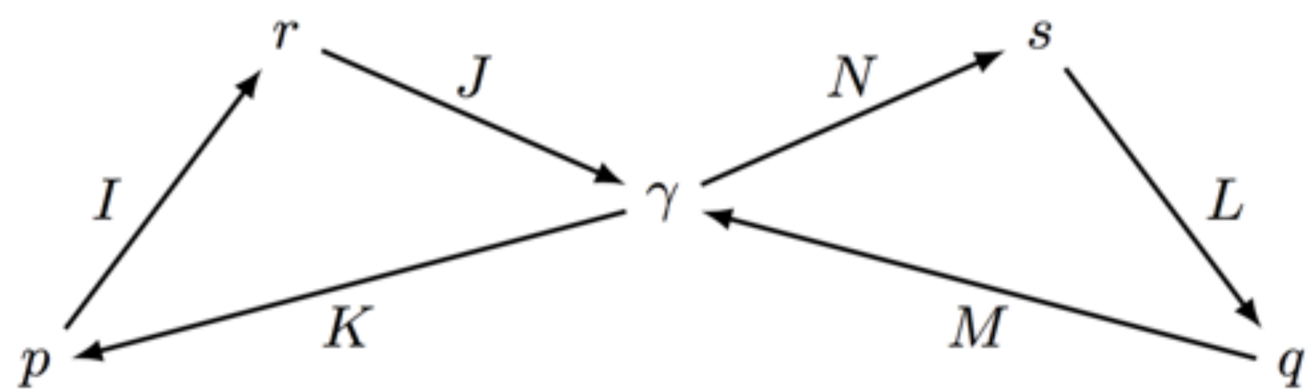
for $x \dashrightarrow y$

definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

(base) $\dfrac{}{x \dashrightarrow x}$ $X_{pp} \supseteq \{0\}$

(transitivity) $\dfrac{x \dashrightarrow y \qquad y \dashrightarrow z}{x \dashrightarrow z}$ $X_{pr} \supseteq X_{pq} + X_{qr}$

(push-pop) $\dfrac{x \dashrightarrow y}{x' \dashrightarrow y'}$ $X_{pq} \supseteq (I + (X_{rs} \cap (J+N)) + L) \cap -(M+K)$



Guideline:
think like state = an integer,
capture all differences y - x,
for $x \dashrightarrow y$

The core problem - no intersections

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

decide, whether its least solution assigns a non-empty set to x_1?

The core problem - no intersections

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1?

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

How to solve the problem in absence of intersections?

The core problem - no intersections

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

decide, whether its least solution assigns a non-empty set to x_1?

How to solve the problem in absence of intersections?

$$\begin{cases} x_1 & = & \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\ x_2 & = & x_1 + \{1\} \cup x_1 + \{-1\} \end{cases}$$

The core problem - no intersections

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

decide, whether its least solution assigns a non-empty set to x_1?

How to solve the problem in absence of intersections?

$$\begin{cases} x_1 & = & \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\ x_2 & = & x_1 + \{1\} \cup x_1 + \{-1\} \end{cases}$$

Decidable in P

The core problem - intersections

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1?

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

The core problem - intersections

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

decide, whether its least solution assigns a non-empty set to x_1?

The problem is undecidable for unlimited intersections.
[Jeż, Okhotin 2010]

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1?

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

decide, whether its least solution assigns a non-empty set to x_1 ?

What about limited intersections: _ ∩ I, for I a finite interval?

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \ldots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

decide, whether its least solution assigns a non-empty set to x_1?

What about limited intersections: _ ∩ I, for I a finite interval?

$$\begin{cases} x_1 & = & \{0\} \ \cup \ x_2 + \{1\} \ \cup \ x_2 + \{-1\} \\ x_2 & = & (x_1 + \{1\} \ \cup \ x_1 + \{-1\}) \ \cap \ \{1\} \end{cases}$$

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1?

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

What about limited intersections: _ ∩ I, for I a finite interval?

$$\begin{cases} x_1 & = & \{0\} \ \cup \ x_2 + \{1\} \ \cup \ x_2 + \{-1\} \\ x_2 & = & x_1 + \{1\} \ \cup \ x_1 + \{-1\} \end{cases}$$

membership problem

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1?

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

What about limited intersections: _ ∩ I, for I a finite interval?

$$\begin{cases} x_1 & = & \{0\} \ \cup \ x_2 + \{1\} \ \cup \ x_2 + \{-1\} \\ x_2 & = & \{1\} \end{cases}$$

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

decide, whether its least solution assigns a non-empty set to x_1?

What about limited intersections: _ ∩ I, for I a finite interval?

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1?

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

What about limited intersections: _ ∩ I, for I a finite interval?

- NP-complete

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \ldots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1 ?

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

What about limited intersections: _ $\cap$ I, for I a finite interval?

- NP-complete
- non-emptiness of constrained definable PDA reduces to the core problem (with exponential blow-up)

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

decide, whether its least solution assigns a non-empty set to x_1?

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

decide, whether its least solution assigns a non-empty set to x_1?

What about _ ∩ I, for I an arbitrary interval?

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union ∪
- point-wise addition +
- limited intersection ∩

decide, whether its least solution assigns a non-empty set to x_1?

What about _ ∩ I, for I an arbitrary interval?

- in EXPTIME, by reduction to 1-BVASS(+ -)

The core problem - limited intersection

Given a systems of equations

$$\begin{cases} x_1 & = & t_1 \\ x_2 & = & t_2 \\ & \dots & \\ x_n & = & t_n \end{cases}$$

- constants {-1}, {0}, {1}
- set union $\cup$
- point-wise addition +
- limited intersection $\cap$

decide, whether its least solution assigns a non-empty set to x_1?

What about _ $\cap$ I, for I an arbitrary interval?

- in EXPTIME, by reduction to 1-BVASS(+ -)
- non-emptiness of definable PDA reduces to the core problem (with exponential blow-up)

Decision procedure

definable PDA

systems of equations
over sets of integers

Decision procedure

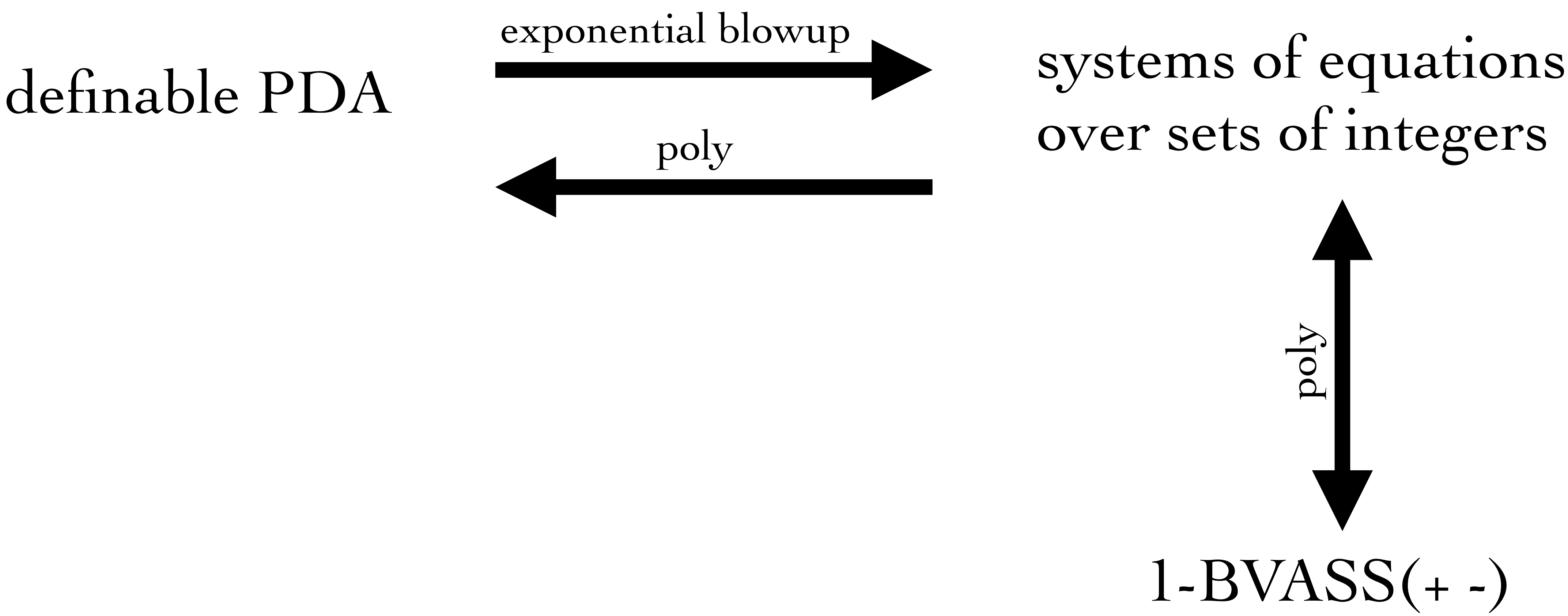
definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

Decision procedure

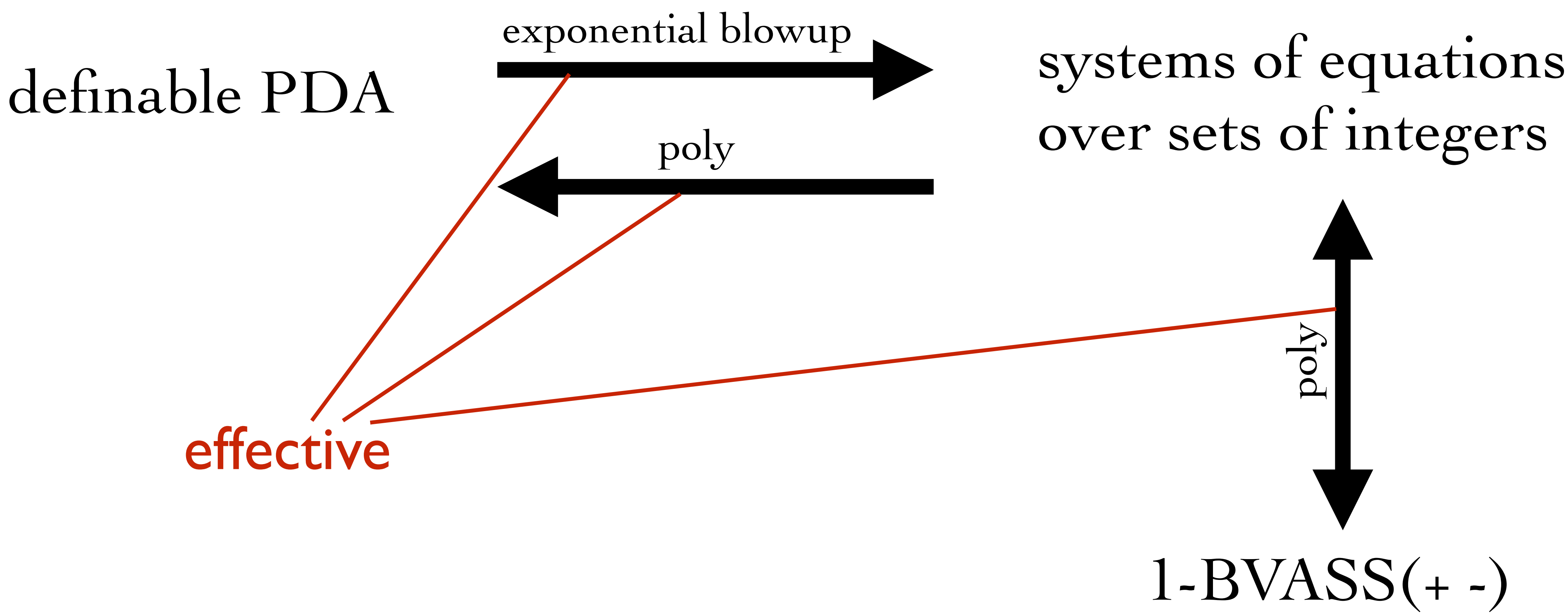
definable PDA $\xrightarrow{\text{exponential blowup}}$ systems of equations over sets of integers

definable PDA $\xleftarrow{\text{poly}}$ systems of equations over sets of integers

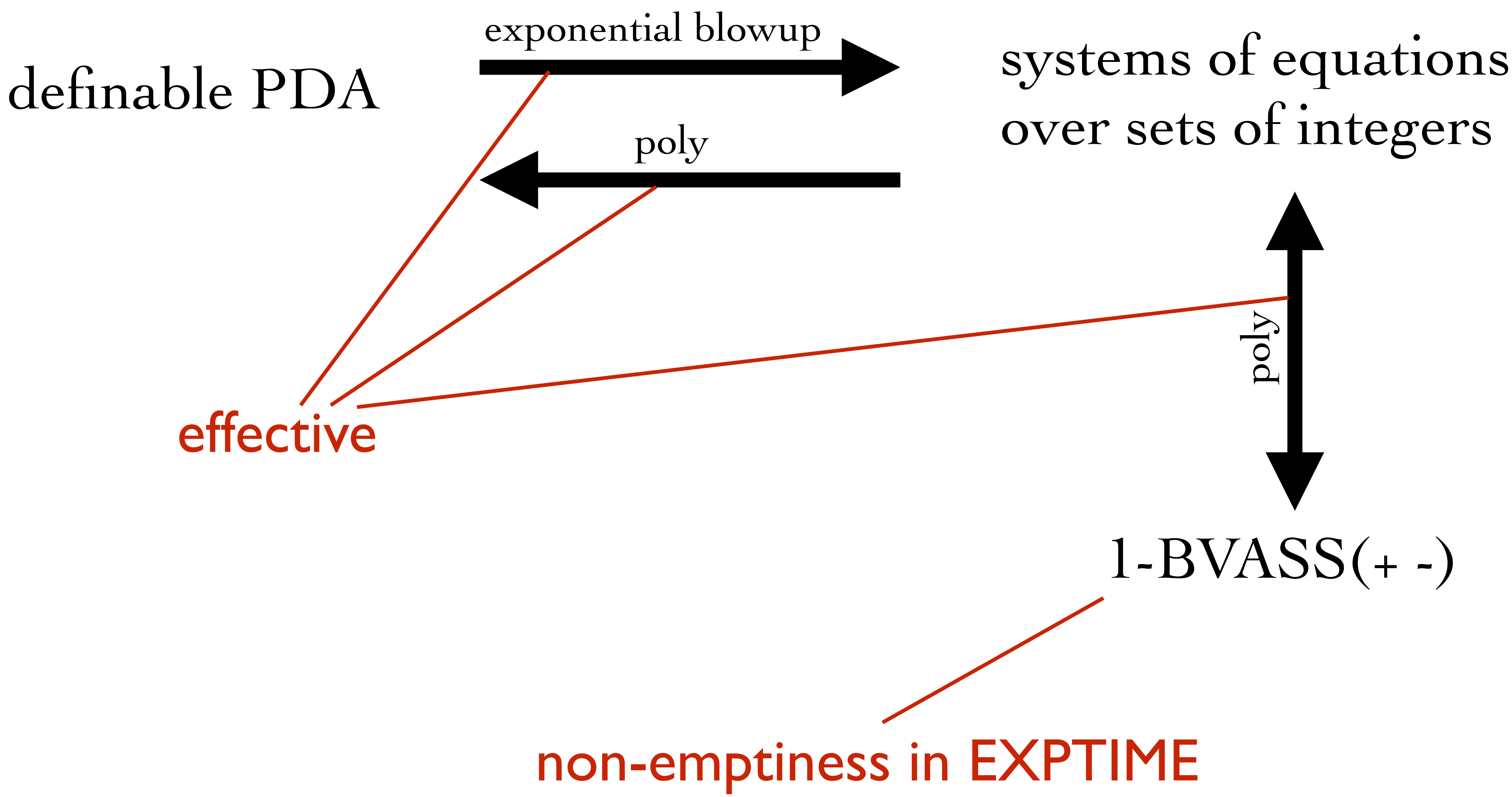
Decision procedure



Decision procedure



Decision procedure



Decision procedure

- Motivation
- Definable NFA
- Definable PDA
- The core problem: equations over sets of integers
- **Branching vector addition systems in dimension 1**

1-BVASS(+ -)

1-BVASS(+ -)

- automaton with 1 non-negative counter

1-BVASS(+ -)

- automaton with 1 non-negative counter
- run is a tree

1-BVASS(+ -)

- automaton with 1 non-negative counter
- run is a tree
- in leaves: initial state with counter=1

1-BVASS(+ -)

- automaton with 1 non-negative counter
- run is a tree
- in leaves: initial state with counter=1
- transition rules:

1-BVASS(+ -)

- automaton with 1 non-negative counter
- run is a tree
- in leaves: initial state with counter=1
- transition rules:

- non-emptiness problem: is there a run with a final state in the root?

Non-emptiness of 1-BVASS(+ -)

Non-emptiness of 1-BVASS(+ -)

Theorem 4:

The non-emptiness problem of 1-BVASS(+ -) is in EXPTIME.

Non-emptiness of 1-BVASS(+ -)

Theorem 4:

The non-emptiness problem of 1-BVASS(+ -) is in EXPTIME.

Proof idea:

Exponentially bounded witness.

Non-emptiness of 1-BVASS(+ -)

Theorem 4:

The non-emptiness problem of 1-BVASS(+ -) is in EXPTIME.

Proof idea:

Exponentially bounded witness.

Complexity gap: PSPACE … EXPTIME

Non-emptiness of 1-BVASS(+ -)

Theorem 4:

The non-emptiness problem of 1-BVASS(+ -) is in EXPTIME.

Proof idea:

Exponentially bounded witness.

Complexity gap: PSPACE … EXPTIME

Theorem: [Goeller, Haase, Lazic, Totzke 2016]

The non-emptiness problem of 1-BVASS(+) is in P (unary encoding).

Definable sets

offer a right setting for timed models of computation, like timed automata, or timed pushdown automata.

Definable PDA

have decidable non-emptiness problem, by reduction to an extension of BVASS in dimension 1.

Definable sets

offer a right setting for timed models of computation, like timed automata, or timed pushdown automata.

Definable PDA

have decidable non-emptiness problem, by reduction to an extension of BVASS in dimension 1.

thank you!